



ECOLE  
POLYTECHNIQUE  
DE BRUXELLES



UNIVERSITÉ LIBRE DE BRUXELLES

TRANH-201 PROJET MULTIDISCIPLINAIRE BA2  
GROUPE 22

---

## Projet Thermostat Intelligent

---

*Auteurs :*  
DENGUIR Anass  
LOUHAJI Najlae  
NOEL Luca  
PHAN Trong Quy

*Superviseur :*  
VERMEIR Aurélien

*Lecteur :*  
VANSUMMEREN Stijn

14 mars 2016

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Conception et aspect technique</b>	<b>5</b>
2.1	Thermostat central . . . . .	5
2.2	Thermostat individuel . . . . .	5
2.3	Environnement de simulation . . . . .	6
2.4	Plan et instruction de montage . . . . .	7
<b>3</b>	<b>Contrôle et communication</b>	<b>8</b>
3.1	Protocole de communication . . . . .	8
3.2	Rôle de l'unité centrale . . . . .	8
3.2.1	Communication . . . . .	8
3.2.2	Gestion des données . . . . .	9
3.2.3	Gestion des erreurs . . . . .	10
<b>4</b>	<b>Régulation de la température</b>	<b>12</b>
4.1	Régulation ON/OFF . . . . .	12
4.2	Régulation PID . . . . .	13
4.2.1	Généralités . . . . .	13
4.2.2	Calibration . . . . .	14
4.3	Indices de qualité . . . . .	18
4.3.1	Critère n°1 : Temps d'atteinte de la température de consigne . . . . .	18
4.3.2	Critère n°2 : Consommation d'énergie . . . . .	19
4.3.3	Critère n°3 : Amplitude des oscillations . . . . .	21
4.3.4	Résultats . . . . .	21
<b>5</b>	<b>Thermostat intelligent</b>	<b>22</b>
5.1	Gestion automatique de la température et prévision des heures d'utilisation . . . . .	22
5.2	Interface d'utilisateur . . . . .	26
<b>6</b>	<b>Fonctionnement du groupe</b>	<b>28</b>
<b>7</b>	<b>Conclusion</b>	<b>31</b>
	<b>Bibliographie</b>	<b>31</b>

<b>A Rapport de recherche bibliographique</b>	<b>34</b>
A.1 Ressources consultées . . . . .	34
A.2 Mots-clés . . . . .	34
A.2.1 Keywords . . . . .	34
A.2.2 Traduction des mots-clés . . . . .	34
A.3 Sources sélectionnées . . . . .	35
A.4 Equations de recherches . . . . .	37
<b>B Information sur le groupe</b>	<b>39</b>

## Abstract

"**Thermostat intelligent.**" : par DENGUIR Anass, LOUHAJI Najlae, NOEL Luca, PHAN Trong Quy, TCHAMKAM NZINGA Marcel.

Cette année, le projet orienté en informatique consiste en la conception d'un thermostat intelligent dont la principale fonction est d'assurer la régulation automatique de la température de plusieurs pièces par l'intermédiaire d'un thermostat central. Les tâches à réaliser consistent en la création d'un thermostat individuel fonctionnant sur base d'un circuit imprimé dénommé Arduino, la réalisation d'un serveur à partir d'un Raspberry Pi transmettant des instructions au thermostat individuel, l'élaboration d'un protocole de communication entre les deux composantes (Arduino et Raspberry Pi), la modélisation de la régulation de la température d'un environnement et l'implémentation d'algorithme rendant le thermostat autonome. L'Arduino et le Raspberry Pi sont fournis par l'Université. La régulation de la température est utilisée pour éviter des oscillations significatives de la température autour d'une consigne afin d'éviter une perte importante en énergie. Il en résultera pour chacune de ces tâches accomplies un thermostat (unité centrale) permettant l'installation de plusieurs thermostats individuels (asservis au thermostat principal).

Mots-clés :thermostat, Arduino, Raspberry Pi, régulation, serveur, communication.

## Abstract

"**Smart thermostat.**" : by DENGUIR Anass, LOUHAJI Najlae, NOEL Luca, PHAN Trong Quy, TCHAMKAM NZINGA Marcel.

This year, the aim of the computer oriented project consists of disigning a smart thermostat which main function is the temperature automatic regulation of several room via a central thermostat. The different tasks ensure the creation of a thin thermostat with a Arduino, the production of a server on a Raspberry Pi sending each instruction to the thin thermostat, the preparation of communication protocol between the two components (Arduino and Raspberry Pi), the temperature regulation modelling and the implementation of algorithms making the thermostat independant. The group responsible of this project was supplied by the university with the Arduino and the Raspberry Pi. The temperature regulation is used to avoid important variation of temperature when the environnement reaches a set temperature. The result of these accomplished tasks is a central thermostat that deals with an undefined number of thin thermostat.

Keywords : thermostat,Arduino,Raspberry Pi, regulation, server, communication

# Chapitre 1

## Introduction

Durant cette année 2015-2016, le projet multidisciplinaire consiste en la conception d'un thermostat multi-zone et intelligent. Ce thermostat est l'association de plusieurs thermostats individuels, placés en des lieux attribués par l'utilisateur, envoyant leurs données enregistrées à un thermostat central. Celui-ci transmettra ses instructions aux thermostats individuels.

Avant le commencement de la conception, un rapport de recherche bibliographique a dû être rendu. Ce rapport a permis au groupe de se baser sur plusieurs ouvrages et de juger la validité des sources. Ce rapport a apporté les bases nécessaires à la compréhension du projet.

Ce projet est décomposé en plusieurs étapes :

- La réalisation du thermostat individuel.
- La conception d'un thermostat central avec algorithme de contrôle réactif.
- L'intégration et validation du thermostat intelligent.
- L'implémentation d'un algorithme de contrôle smart, c'est-à-dire optimiser le comportement du thermostat.

Il a été décidé de diviser ce rapport de mi-parcours en plusieurs parties décrivant le déroulement du projet :

- la conception, câblage et les aspects techniques du thermostat individuel
- la gestion de la communication entre l'Arduino et le Raspberry Pi
- la régulation de la température
- la partie intelligente
- le fonctionnement de groupe, composé de la description de l'organisation du groupe

Cette division en plusieurs chapitres permet une vision globale du projet ainsi qu'une lecture plus agréable pour le lecteur. Ainsi, nous vous souhaitons une bonne lecture.

# Chapitre 2

## Conception et aspect technique

Dans ce chapitre, nous allons décrire brièvement les aspects physiques du prototype ainsi que les différents capteurs utilisés.

### 2.1 Thermostat central

Le thermostat central est constitué d'un Raspberry Pi 2 qui est un micro-ordinateur de faible coût. L'avantage de son utilisation est son faible prix et sa faible consommation électrique. Il est doté d'un dongle WiFi afin de générer un réseau WiFi et permettre une communication sans fil avec les différents thermostats satellites. C'est sur ce dernier qu'est implémenté l'algorithme de contrôle de la température et de gestion des thermostats individuels.

### 2.2 Thermostat individuel

Le thermostat individuel est constitué d'un Arduino Nano et est lié au thermostat central avec lequel il communique constamment via un protocole HTTP grâce à un module WiFi ESP8266 pré-configuré pour la transmission de requêtes. Il est doté de deux capteurs, un thermistor et un capteur infrarouge, qui permettent respectivement de relever la température et la présence de personne dans une pièce. Concernant leur fonctionnement, le capteur infrarouge envoie un signal électrique chaque fois qu'il détecte un mouvement tandis que le thermistor est une résistance qui varie en fonction de la température.



FIGURE 2.1 – Détecteur de présence



FIGURE 2.2 – Thermistance

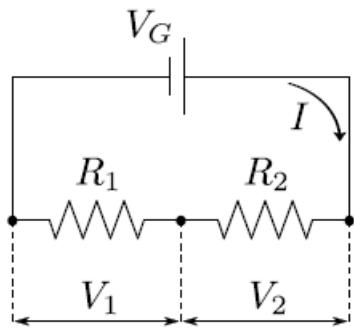
Afin de récolter la température à partir du thermistor, nous avons recouru à une résistance de référence de  $10\text{k}\Omega$  et mis en place un pont diviseur de tension pour calculer la résistance du thermistor [11] et ainsi trouver la température grâce à la relation d'interpolation de Steinhart-Hart ci-dessous (2.1). Dans la relation ci-dessous, A, B, C, D sont des constantes qui sont relatives au modèle de thermistance employé. Ces valeurs sont présentées dans le tableau des paramètres fournis par la compagnie Vishay[17]

### Relation de Steinhart-Hart :

$$\frac{1}{T(R)} = A + B \cdot \ln\left(\frac{R}{R_{ref}}\right) + C \cdot \ln\left(\frac{R}{R_{ref}}\right)^2 + D \cdot \ln\left(\frac{R}{R_{ref}}\right)^3 \quad (2.1)$$

La thermistance est placée en série avec la résistance de  $10\text{k}\Omega$  dans le circuit. On calculera donc la résistance de la thermistance à partir de la mesure de tension aux bornes de la résistance de référence par analyse d'un diviseur de tension. Afin de convertir les données reçues en bits par l'Arduino (entre 0 et 1023) en une tension, il suffit de la diviser par la valeur maximale en bits, 1023, et de multiplier le tout par la tension du générateur, en l'occurrence 5 volt. Nous pouvons ensuite calculer la résistance de la thermistance et obtenir la température avec la relation de Steinhart-Hart.

### Diviseur de tension (issu du syllabus de laboratoire de M. Haelterman [10]) :



Lorsque deux résistances  $R_1$  et  $R_2$  placées en série sont connectées à un générateur, la tension  $V_G$  fournie par celui-ci se divise en :

- une fraction  $V_1 = R_1 I$  aux bornes de  $R_1$  ;
- une fraction  $V_2 = R_2 I$  aux bornes de  $R_2$ .

La d.d.p. totale est la somme des deux d.d.p. partielles :  $V_G = V_1 + V_2 = (R_1 + R_2)I$ , et on peut écrire :

$$V_1 = \frac{R_1}{R_1 + R_2} V_G \quad \text{et} \quad V_2 = V_G - V_1. \quad (1.4)$$

## 2.3 Environnement de simulation

L'environnement de simulation a été fourni par les organisateurs du projet. Celui-ci est constitué d'une résistance chauffante qui fait office de radiateur. L'actionnement de la valve est alors contrôlé via la transmission d'un courant modulé en amplitude (PWM) [1] par le thermostat individuel. Il comporte également une entrée pour le branchement du capteur de température extérieur et permet d'alimenter le thermostat individuel.

## 2.4 Plan et instruction de montage

Voici le plan de câblage du thermostat individuel avec l'environnement de simulation (bloc de couleur brun).

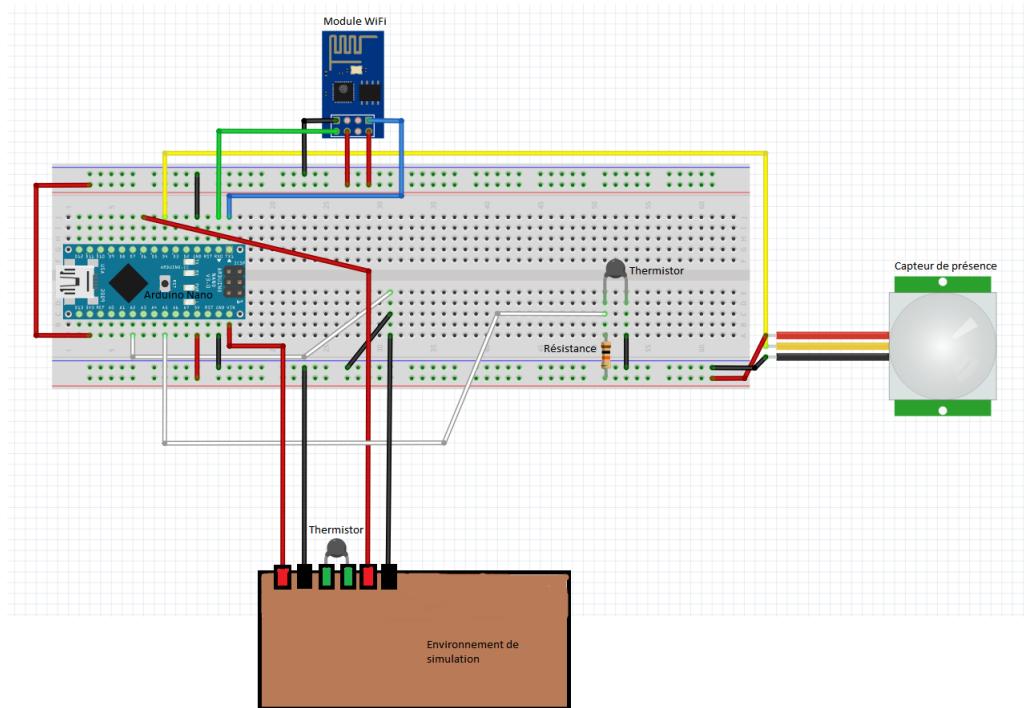


FIGURE 2.3 – Câblage de l'Arduino avec l'environnement de simulation

### Instruction de montage :

1. Brancher l'entrée RX du module WiFi avec la pin TX de l'Arduino ainsi que la sortie TX du module WiFi avec la pin RX de l'Arduino
2. Connecter le module WiFi à une source de tension de 3.3V et refermer le circuit en connectant la borne de sortie à la masse
3. Alimenter en 5V le capteur de présence et connecter le capteur à la masse
4. Connecter la 3ème entrée du capteur de présence à une pin digitale de l'Arduino
5. Brancher la thermistance en série avec la résistance de  $10k\Omega$  et alimenter en 5V
6. Câbler la vanne (en PWM)
7. Câbler l'alimentation de l'Arduino

# Chapitre 3

## Contrôle et communication

### 3.1 Protocole de communication

Afin de garantir une communication sans fil efficace entre l'unité centrale et les thermostats satellites, il nous a été demandé d'employer des requêtes HTTP [4] pour la transmission d'informations. Ces requêtes HTTP permettent différentes actions tel que l'enregistrement des thermostats individuels au sein de l'unité centrale, la transmission de donnée issues des différents capteurs ainsi que le contrôle de la vanne thermostatique. Dans le cadre de ce projet, nous avons eu recours à trois méthodes HTTP :

- PUT : lors de l'enregistrement
- GET : lors de la récolte des informations fournies par les capteurs
- POST : lors de la mise en service de la vanne thermostatique

Dès l'initialisation d'un thermostat individuel, son adresse IP et son port sont envoyés à l'unité centrale, dont l'adresse IP est fixe, afin que ce dernier puisse ensuite les utiliser pour les interroger au moyen des requêtes HTTP.

### 3.2 Rôle de l'unité centrale

#### 3.2.1 Communication

L'unité centrale a pour principal objectif la gestion de la communication avec les différents thermostats individuels. Elle se fait via un réseau Wifi qui a été préalablement configuré et la communication client-serveur respecte un protocole bien précis décrit dans la section précédente. D'une part les thermostats satellites transmettent les informations récoltées par leurs différents capteurs, d'autre part l'unité centrale transmet les données nécessaires à la régulation du chauffage. Pour ce faire, le thermostat central doit être doté d'algorithmes de contrôle qui ont été implémentés en Python sur le Raspberry Pi. Les différents algorithmes de contrôles utilisés seront décrits en détail ultérieurement.

La communication entre le thermostat central et les thermostats satellites (voir figure 3.1) se fait de la manière suivante :

1. Tout d'abord l'unité centrale interroge chaque thermostat individuel au moyen d'une requête PUT et les invite à s'enregistrer dans sa base de données. C'est pourquoi chaque

Arduino doit être configuré afin de pouvoir répondre à cette demande d'enregistrement. Il s'agit d'une tâche de fond qui s'exécute en permanence.

2. Une fois que les adresses IP et ports des thermostats individuels nous sont fournis lors de l'enregistrement de ces derniers, il nous suffit dès lors de les parcourir au moyen d'une boucle *for* afin de toutes les consulter. Cependant une connexion doit au préalable être établie avec chaque thermostat. Pour ce faire, nous avons implémenté une classe *Sensor* qui va garantir l'établissement de cette connexion grâce aux différentes méthodes de la librairie *requests*. A chaque thermostat satellite est associé un objet de cette classe dont les méthodes vont permettre un accès en lecture (méthode *get*) et en écriture (méthode *set*) aux URL spécifiées. Concrètement, les accesseurs en lecture vont permettre à l'unité centrale de lire l'état des différents capteurs de chaque thermostat individuel tandis que les accesseurs en écriture vont permettre de modifier l'état de la vanne thermostatique.
3. Enfin, le thermostat central sonde chaque appareil enregistré et leur demande plusieurs informations telles que la température actuelle, le pourcentage d'utilisation de la vanne thermostatique et l'éventuelle détection d'une présence. Afin d'assurer une bonne gestion des informations captées, celles-ci transitent sur des chemins bien distincts : ce sont les URL.

Une URL est le chemin d'accès à une ressource du serveur questionné, elle présente une structure semblable à la suivante : *protocole://adresse ip :port/ressource*

Le premier mot indique que la requête qui est faite au serveur respecte un protocole bien précis, dans notre cas il s'agit du protocole HTTP. Ensuite, il est nécessaire de spécifier l'appareil interrogé, ceci se fait en mentionnant son adresse IP ainsi que le port sur lequel il est connecté. Enfin tout ce qui s'ensuit représente le chemin d'accès à la ressource recherchée. Celle-ci peut contenir des informations enregistrées sous différents formats (HTML, text, JSON, etc). Par exemple, pour importer la température captée par un Arduino, il a fallu parcourir l'URL suivante : *http://ip-arduino :port-arduino/temperature*

De plus, afin d'assurer une communication simultanée entre l'unité centrale et les différents thermostats, nous avons utilisé le module *threading* de Python qui va permettre au Raspberry Pi de communiquer avec tous les arduinos en même temps. Cette fonctionnalité voit son intérêt lorsque le nombre de thermostats satellites devient important. En effet, on peut aisément imaginer que plus ce nombre s'accroît, plus le temps de séparation entre deux sondages d'un même thermostat est long. Ceci peut devenir problématique dans le cadre d'une régulation qui nécessite une prise de connaissance régulière de la température.

### 3.2.2 Gestion des données

L'unité centrale assure également le stockage de données, celles-ci sont utiles pour la prise de mesures expérimentales et l'implémentation d'algorithmes de contrôle intelligents se basant sur des données prélevées précédemment.

A ce stade, deux moyens de stockage d'informations se présentent :

- L'utilisation de fichiers
- L'implémentation d'une base de données

Les fichiers sont faciles à utiliser et peuvent être modifiés manuellement en cas de besoin, cependant on ne peut pas enregistrer suffisamment de données dans un fichier. Les bases de

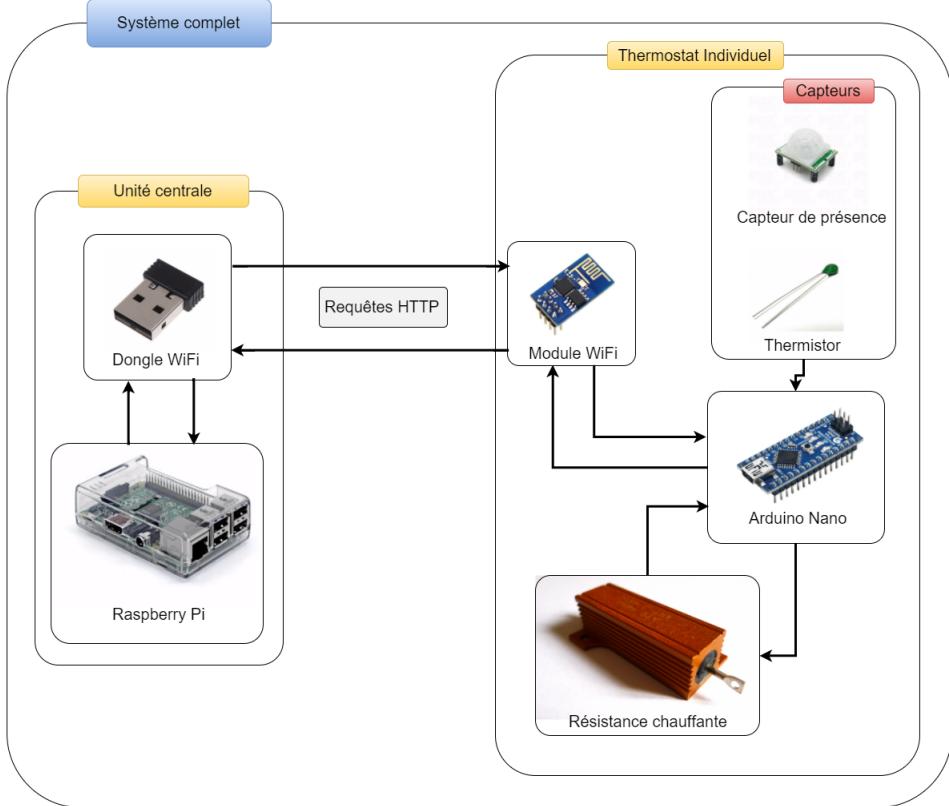


FIGURE 3.1 – Diagramme de communication

données ont beaucoup plus de capacité de stockage et présentent une structure de répartition par tables plus adaptée à la prise de mesures de données numériques. De plus, elles permettent une gestion des informations optimales grâce à ces fonctionnalités de tris de données. C'est pourquoi nous avons préféré la seconde option. La base de données a été implémentée en utilisant *SQLite* avec la bibliothèque *sqlite3* de Python. Ce module est facile d'utilisation et a l'avantage d'être pré-installé sur Python. D'autres bases de données telles que *MySQL* ou *Oracle* auraient pu convenir mais elles ne présentaient pas de facilité d'accès de *SQLite* [15].

La base de donnée a été implémentée de manière à assurer deux tâches :

- Elle va permettre d'une part le stockage des données de l'utilisateur. Ces données seront utiles à l'implémentation d'une intelligence artificielle se basant sur les statistiques de ce dernier. Cet aspect du projet est décrit en détail dans le chapitre 5.
- D'autre part ces informations seront utilisées afin d'afficher les dernières données fournies par chaque thermostat dans une dashboard consultable par l'utilisateur depuis un navigateur web.

### 3.2.3 Gestion des erreurs

Afin de faciliter le codage de la communication du Raspberry Pi, celui-ci a d'abord été testé avec l'environnement de simulation, en effet celui ci a été fourni de telle sorte qu'il soit déjà opérationnel pour la communication sans fil avec notre prototype. Ainsi une éventuelle

erreur de communication ne pouvait provenir que du code Python. En ce qui concerne les erreurs liées au code Python, elles ont été gérées grâce au couple d'instructions *try-except*. Cette instruction a l'avantage de continuer l'exécution du programme malgré la survenance d'une erreur dans l'algorithme de contrôle. La commande *except* permet de sauter les lignes de code sujet à erreur et d'exécuter ce qui suit ce mot clef. Dans notre situation, la majorité des erreurs apparaissent lors de l'exécution de requêtes. En effet, le matériel utilisé dans ce projet ne permet pas une communication optimale entre l'unité centrale et les thermostats satellites. Afin de gérer ces erreurs nous avons opté pour la mise en place d'un système de timeout directement disponible à partir de la fonction *get* du module *requests* : Si la requête ne s'effectue pas avant un certain nombre de secondes, celle-ci renvoie *None* et l'exécution du code reprend son cours.

# Chapitre 4

## Régulation de la température

Le but du projet étant de pouvoir chauffer une pièce, il est donc important de pouvoir générer un algorithme de contrôle accomplissant cette tâche. Ce dernier sera implémenté dans le thermostat central et via le protocole de communication ,des ordres pourront être envoyés aux thermostats individuels. Une fois les commandes reçues, la vanne pourra être actionnée selon la température de consigne de la zone. L'implémentation de l'algorithme de contrôle se déroule en plusieurs étapes. Il s'agit de mettre en place un codage basé sur une détection réactive. Pour ce faire, nous débutons avec une régulation on/off suivie de l'étude de la conception d'un P.I.D.

### 4.1 Régulation ON/OFF

La régulation on/off est une simple méthode permettant d'actionner la vanne à 100% dans le cas où la température ambiante est en dessous de la température de consigne . Elle désactive l'ouverture de la vanne lorsque la température ambiante vaut ou dépasse la consigne.  
Voici l'allure des courbes de température obtenues lors de nos expériences (figure 4.1) :

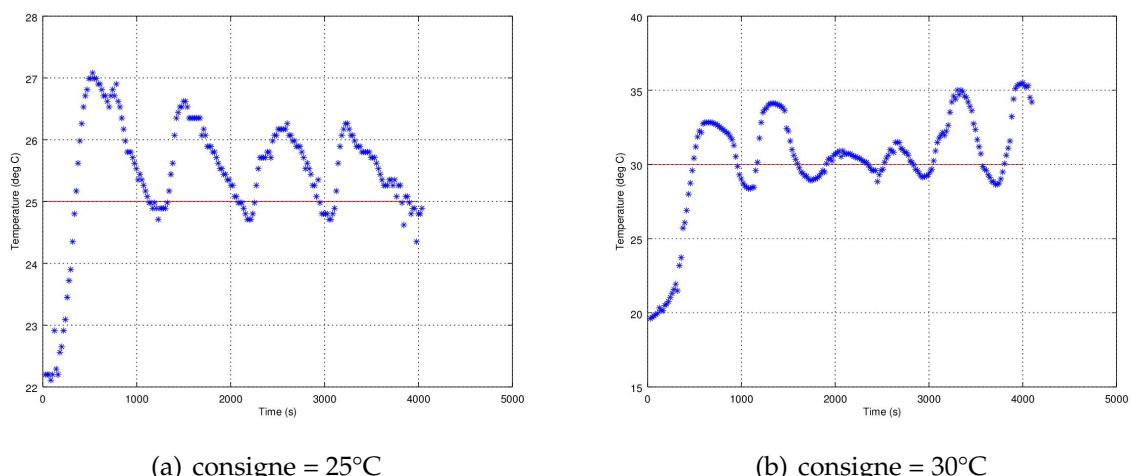


FIGURE 4.1 – Mesures pour l'algorithme On/Off pour différentes températures de consigne

Sur ces graphes, on observe de fortes oscillations autour de la température de consigne. Cela est dû au fait que le système a une inertie dont l'algorithme on/off ne tient pas compte. Ce phénomène est fortement visible sur la courbe verte de la figure 5.1. Dès que la température de consigne est dépassée, la vanne va à 0 mais l'inertie du système fait que la température continue quand même à augmenter.

## 4.2 Régulation PID

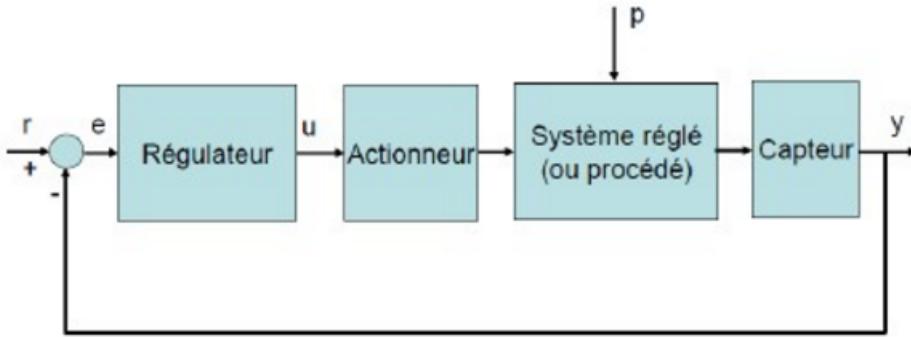
### 4.2.1 Généralités

La régulation P.I.D tente d'approcher la valeur de consigne au plus juste tout en faisant en sorte que les pertes de chaleur de la zone à chauffer soient compensées en analysant régulièrement le système et en ajustant systématiquement la commande en fonction de la valeur de consigne (voir figure 4.2).

La méthode PID (proportionnel, intégrale, dérivée) se décompose en 3 fonctions (f4.1). Tout d'abord, la partie proportionnelle qui dépend de l'écart de température entre la température ambiante et la consigne. Cet écart portera le nom d'erreur. En effet, plus l'erreur est grande plus le pourcentage d'ouverture de la vanne augmente. Considérons une consigne fixée à 20°C et une température ambiante de 15°C, la vanne est en premier lieu activée à 100%. Seulement, lorsque l'erreur diminue et que la température est à 19°C, la vanne n'est ouverte plus qu'à 30%. Le but n'étant de jamais vraiment atteindre la valeur consigne car si la vanne se ferme la température baisse. Dans la formule, l'erreur est multipliée par un coefficient  $K_p$ .

La seconde branche de l'équation est une intégrale qui consiste à faire la somme des erreurs au cours du temps. Plus on chauffe, plus l'erreur globale augmente et plus on continue de chauffer. Mais dans le cas où la température dépasse la consigne, l'erreur globale diminue ce qui permet de décélérer la commande. Cette composante est additionnée à la partie proportionnelle car elle permet de pouvoir annuler l'erreur restante (due à la composante proportionnelle) de sorte à ce que la température soit égale à la consigne.

La dernière composante est une dérivée. Elle peut se calculer en considérant l'erreur courante et l'erreur précédente divisées par le temps séparant ces deux erreurs. Le but de cette dérivée est de pouvoir freiner la commande dans le cas où l'augmentation de température est trop rapide. Ceci permettra de pouvoir atteindre au mieux la valeur consigne en évitant le plus d'oscillations possibles.



r : signal de référence (à suivre)

y : signal de mesure

u : commande

p : perturbation

e : erreur de réglage

FIGURE 4.2 – Schéma régulation présenté lors du séminaire de Laurent Catoire

$$CO = K_p \cdot erreur(t) + K_i \cdot \int_{t_i}^{t_f} erreur(t) dt + K_d \cdot \frac{derreur(t)}{dt} \quad (4.1)$$

où CO = commande (pourcentage d'ouverture de la vanne)

Ce modèle est beaucoup plus rigoureux que le précédent. Son avantage réside dans le fait qu'il assure une régulation sans pour autant imposer la connaissance des paramètres physiques du système. Cependant il faut encore déterminer les trois constantes  $K_p$ ,  $K_i$  et  $K_d$  : Celles-ci peuvent se trouver expérimentalement. La principale difficulté de l'implémentation du régulateur PID est le fait que les constantes varient en fonction des différents paramètres physiques du système comme la conductivité thermique du matériau isolant, la température initiale de la pièce, etc. Ainsi les constantes  $K_p$ ,  $K_i$ ,  $K_d$  ne sont pas vraiment des constantes dans le sens où elles n'auront pas forcément les mêmes valeurs d'une période de la journée à une autre ou d'un milieu expérimental à un autre.

Il est donc nécessaire d'introduire à notre algorithme de régulation une phase de calibration dont le but sera de déterminer les valeurs des 3 constantes. Pour ce faire, nous avons eu recours à la méthode de Ziegler-Nichols[2]. Celle-ci est décrite dans la section suivante.

### 4.2.2 Calibration

La méthode de Ziegler-Nichols est une méthode graphique qui consiste à maintenir la vanne allumée à une valeur constante pendant un intervalle de temps suffisamment long. Une fois la courbe de l'évolution de la température en fonction du temps obtenue, il suffit de trouver le point présentant la plus grande pente. La courbe obtenue après ce processus est semblable à la suivante : 4.3 :

	$K_p$	$K_i$	$K_d$
PID	$\frac{12}{aL}$	$\frac{100 \cdot K_p}{2 \cdot L}$	$\frac{K_p \cdot L}{2000}$

TABLE 4.1 – Tableau des constantes du PID

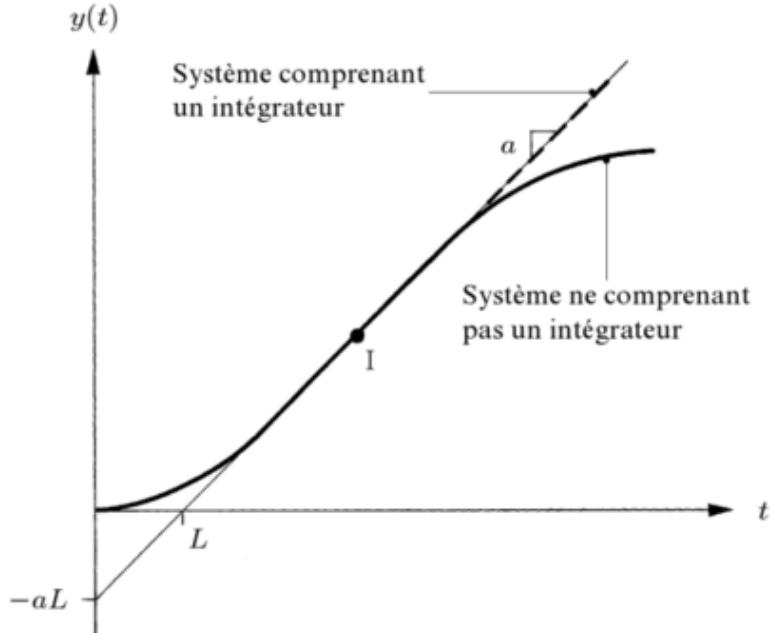


FIGURE 4.3 – Méthode de Ziegler-Nichols en théorie tirée de l'ouvrage : [2]

Après avoir trouvé ce point particulier (il s'agit du point  $I$ ), il suffit de trouver le point d'intersection des droites suivantes :

$$\begin{cases} y(t) = y_I + a(t - t_I) \\ y(t) = T_0 \end{cases}$$

où :

- $y(t)$  représente la température en fonction du temps
- $T_0$  représente la température initiale du processus de chauffe (il s'agit de l'axe des abscisses dans la figure 4.3)
- $a$  est la plus grande pente du graphique pondérée par la valeur de la vanne (comprise entre 0 et 100)

La solution de ce système d'équations nous renseigne sur le point  $L$  qui représente le temps de réaction de chauffage due à l'inertie du système. Une fois les variables  $a$  et  $L$  trouvées, Ziegler-Nichols nous proposent des formules dépendantes de ces deux paramètres qui nous permettront de déterminer les constantes  $K_p$ ,  $K_i$  et  $K_d$ . Cependant nous nous sommes permis d'adapter ces formules aux expériences faites sur notre prototype compte-tenu de la nature empirique de ces relations. Les valeurs obtenues sont représentées dans le tableau 4.1.

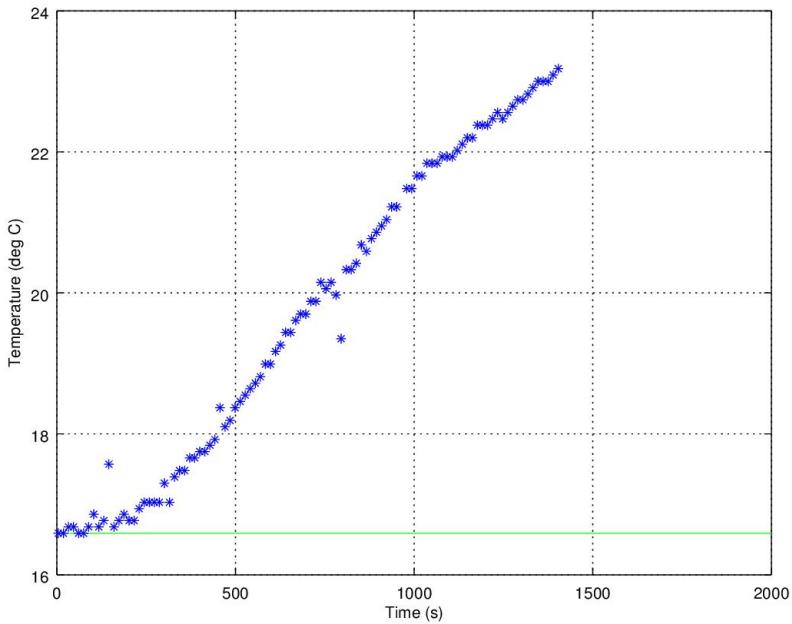


FIGURE 4.4 – Méthode de Ziegler-Nichols en pratique

En pratique, lors de la phase de calibration, nous utilisons l’analyse numérique afin de déterminer le point  $I$ . En effet il suffit de se rappeler que :

$$\frac{dy_i}{dt} = \frac{y_{i+1} - y_{i-1}}{2 \cdot h} + O(h^2) \quad (4.2)$$

$$a = \max\left(\frac{dy_i}{dt}\right) \quad (4.3)$$

où  $h$  est le pas d’intégration, typiquement il s’agit du temps séparant la mesure  $i$  et  $i+1$ .

Cependant cette méthode présente quelques inconvénients :

- La phase de calibration peut durer de 15 à 20 minutes.
  - La méthode utilisée est très sensible au bruit de mesures. Ceci est illustré dans la figure 4.4. On observe que l’allure de la courbe de calibration que nous avons obtenu est la même que celle présentée théoriquement, néanmoins on peut remarquer la présence de mesures dites parasites situées en dehors de la courbe. A premier abord, ceci paraît négligeable, toutefois on remarque que ces bruits de mesures peuvent perturber grandement les valeurs des constantes lorsque l’on analyse les équations (5.2) et (5.3). En effet, le point du graphique présentant la plus grande pente sera probablement un point situé au voisinage de ces mesures parasites alors que ce n’est pas forcément le cas.
- Afin d’éviter ce phénomène, nous avons dû implémenté un algorithme qui détecte les valeurs aberrantes de la série statistique obtenue. Ce-dernier éliminera tous les points dont l’accroissement est supérieur à une valeur limite définie arbitrairement.

Une fois les constantes déterminées, on s’attend à obtenir une courbe de la régulation PID semblable à celle représentée sur la figure 4.5.

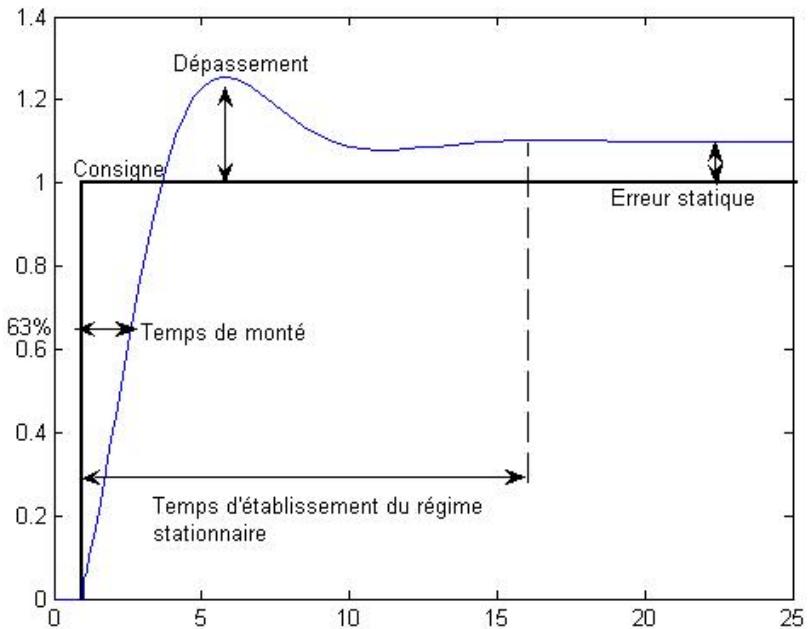


FIGURE 4.5 – P.I.D : Image tirée de Wikipédia

Les courbes obtenues expérimentalement (voir figure 4.6) ont une allure proche de celle recherchée.

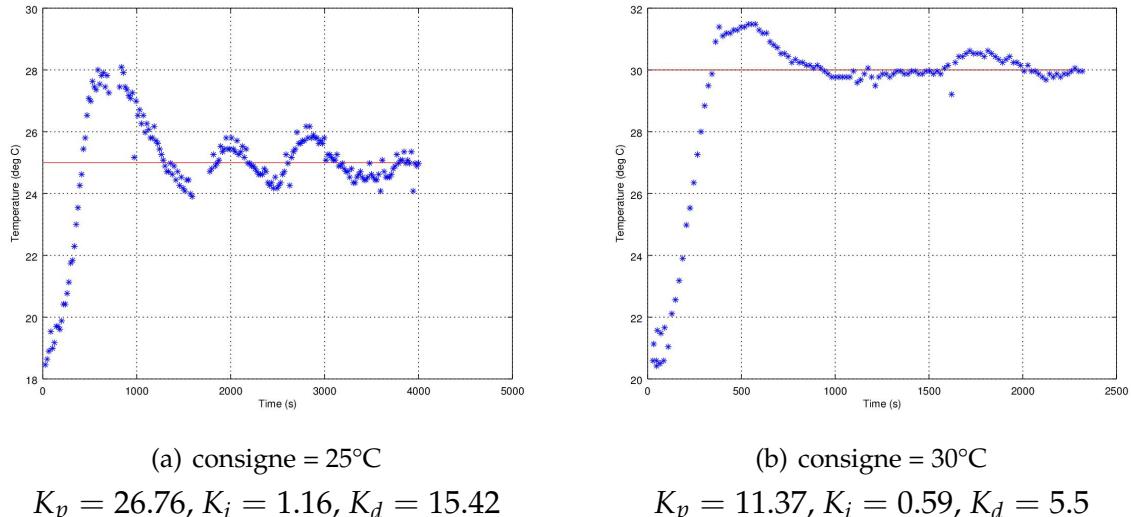


FIGURE 4.6 – Mesures du PID pour différentes températures de consigne

## 4.3 Indices de qualité

Après avoir discuté les différents algorithmes implémentés pour la régulation de la température, il est nécessaire d'introduire un indice de qualité. Il s'agit d'un nombre compris entre 0 et 1 qui jugera l'efficacité de l'algorithme implémenté. Différents facteurs seront pris en compte lors de l'évaluation des différentes méthodes de régulation, à savoir :

1. La vitesse à laquelle la température atteint sa valeur de consigne
2. La consommation d'énergie du chauffage
3. L'amplitude moyenne des oscillations autour de la température de consigne

Une fois les critères d'évaluation de l'algorithme établis, il est naturel de leur attribuer une pondération qui les classera par ordre d'importance, ce choix est tout à fait arbitraire, d'ailleurs les critères mentionnés ci-dessus le sont également.

Nous avons décidé de pondérer chaque point de la manière suivante :

$$i = \frac{1}{6} \cdot c_1 + \frac{1}{2} \cdot c_2 + \frac{1}{3} \cdot c_3 \quad (4.4)$$

où :

- $i$  est l'indice de qualité avec :  $0 \leq i \leq 1$
- $c_1, c_2, c_3$  sont les trois critères d'évaluation cités ci-dessus avec :  $0 \leq c_k \leq 1 (k = 1, 2, 3)$

### 4.3.1 Critère n°1 : Temps d'atteinte de la température de consigne

Afin de définir le niveau de vitesse de chauffage, il est nécessaire d'introduire une valeur de celle-ci que l'on considérera comme optimale. Il s'agit d'une valeur qui ne peut être atteinte que si la vanne est allumée à 100% dès le début de l'expérience. Il est à noter que les valeurs obtenues ne sont cohérentes que si les différentes mesures sont effectuées dans de même conditions expérimentales.

On définit alors :

$$c_1 = \frac{\text{temps}_{optimal}}{\text{temps}_{mesure}} \quad (4.5)$$

Compte tenu de la définition du temps optimal, on déduit aisément que, pour l'algorithme *On/Off*,  $c_1 = 1$ , c'est-à-dire la valeur maximale.

Les graphes ci-dessous (4.7) représentent les temps mis par chaque algorithme pour atteindre la température de consigne :

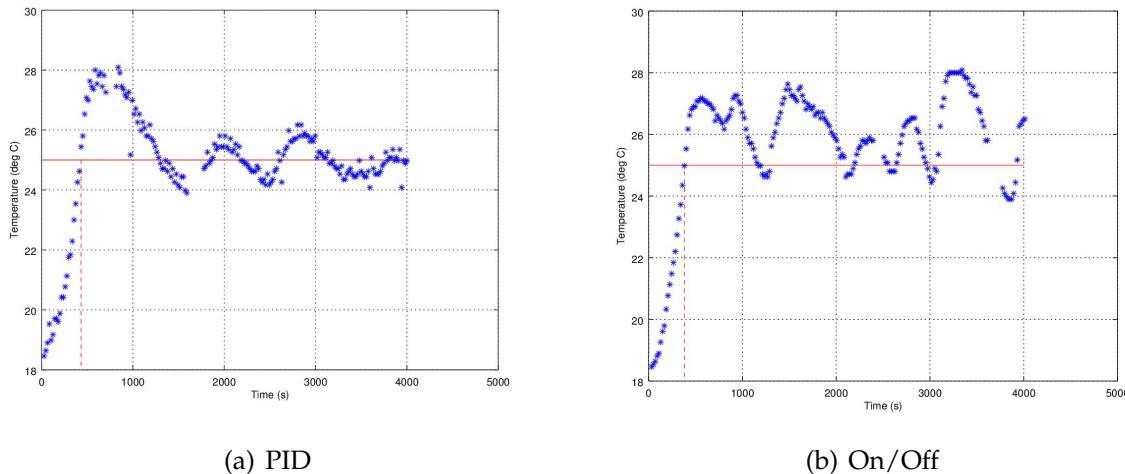


FIGURE 4.7 – Graphes de comparaison entre la régulation on/off et PID

On constate comme attendu que le régulateur On/Off est plus rapide que le PID. En effet, on obtient un temps de 432 secondes pour le PID contre 380 secondes pour le second. On a alors compte tenu de l'équation (5.5) :

$$\begin{cases} c_{1,on/off} = 1 \\ c_{1,pid} = \frac{380}{432} \approx 0.88 \end{cases}$$

#### 4.3.2 Critère n°2 : Consommation d'énergie

Un autre critère que nous jugeons pertinent pour l'évaluation de nos algorithmes de chauffage est l'énergie totale consommée après chaque processus de chauffe. A priori ce critère n'est pas facile à déterminer car il dépend de plusieurs paramètres physiques comme illustré dans la formule ci-dessous :

$$Q_i = m \cdot C_p \cdot (T_i - T_0) \quad (4.6)$$

Cependant on peut aisément s'affranchir des paramètres physiques  $m$  et  $C_p$  car les milieux expérimentaux restent les mêmes d'une expérience à l'autre. Ainsi, ces constantes physiques propres aux conditions expérimentales ne seront plus objets à comparaison.

Toutefois il est utile de se rappeler que l'énergie dissipée est proportionnelle à la différence entre la température à la  $i^{eme}$  itération  $T_i$  et la température initiale  $T_0$ . De plus, en se rappelant que :

$$Q = \sum_i Q_i \quad (4.7)$$

on peut aisément ramener la comparaison de l'énergie dissipée de nos deux algorithmes à celle de l'aire sous la courbe de chacun de leur graphe.

Afin de déterminer l'aire sous la courbe de chaque graphe, nous avons eu recours à la méthode des trapèzes. Celle-ci consiste à découper le graphique en de nombreux trapèzes arbitrairement petits. La somme des aires de chaque trapèze approxime alors assez bien l'aire sous la courbe étudiée.

En outre il est nécessaire d'introduire une aire dite optimale pour faciliter la définition du critère  $c_2$ . Celle-ci sera l'aire du rectangle défini ci-dessous :

$$Aire_{optimale} = (t_{fin} - t_{montee}) \cdot (T_{consigne} - T_0) \quad (4.8)$$

où :

- $t_{montee}$  et  $t_{fin}$  sont respectivement le temps optimal de montée jusqu'à la température de consigne  $T_{consigne}$  et le temps total de l'expérience
- $T_0$  est la température initiale à laquelle débute l'expérience

Similairement au critère  $c_1$ , on définira le second critère de la façon suivante :

$$c_2 = \frac{Aire_{optimal}}{Aire_{mesure}} \quad (4.9)$$

Où  $Aire_{optimal}$  est déjà connue, elle vaut d'après sa définition (4.8) :

$$Aire_{optimale} = (4000 - 380) \cdot (25 - 18.5) = 23530 \quad (4.10)$$

Voici les aires obtenues pour les deux algorithmes étudiés, il s'agit de la surface délimitée en jaune dans les graphes 4.8

$$\begin{cases} Aire_{pid} = 25652 \\ Aire_{on/off} = 28700 \end{cases}$$

Nous obtenons ainsi les valeurs suivantes pour le second critère :

$$\begin{cases} c_{2,pid} = \frac{23530}{25652} \approx 0.92 \\ c_{2,on/off} = \frac{23530}{28700} \approx 0.82 \end{cases}$$

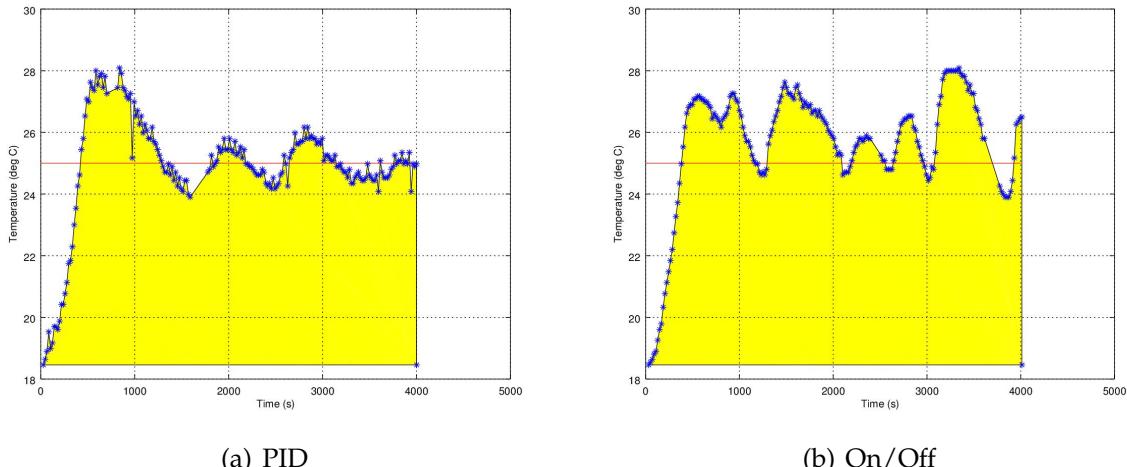


FIGURE 4.8 – Graphes de comparaison entre la régulation On/Off et PID

### 4.3.3 Critère n°3 : Amplitude des oscillations

Notre dernier critère d'évaluation consiste à mesurer la stabilité de chaque algorithme de régulation. Pour ce faire nous calculons la somme des amplitudes des oscillations de la température autour de la valeur de consigne. Plus l'amplitude sera faible, plus la valeur de  $c_3$  sera proche de 1. A contrario une grande valeur de cette amplitude sera synonyme d'instabilité et  $c_3$  se rapprochera de 0. Voici la définition choisie du troisième critère de comparaison :

$$c_3 = \frac{1}{|1 + amplitude|} \quad (4.11)$$

La valeur absolue intervenant dans la formule (4.11) assure que l'amplitude sera positive (une amplitude peut être négative si l'oscillation se fait en-dessous de la valeur de consigne). Des trois critères discutés jusqu'à présent, il s'agit du plus exigeant car il présente de faibles valeurs pour les deux algorithmes.

Les résultats obtenus pour ce dernier critère de comparaison sont les suivants :

$$\begin{cases} c_{3,pid} = \frac{1}{|1+2.47|} \approx 0.29 \\ c_{3,on/off} = \frac{1}{|1+9.28|} \approx 0.1 \end{cases}$$

### 4.3.4 Résultats

Après avoir calculé nos différents critères de comparaison, nous pouvons en déduire la valeur des indices de qualité pour chaque algorithme de régulation. En appliquant la formule (4.4), on obtient les valeurs suivantes :

$$\begin{cases} i_{pid} \approx 0.7 \\ i_{on/off} \approx 0.61 \end{cases}$$

On constate que la régulation PID est plus performante que la régulation On/Off. Cependant, comme mentionné précédemment, l'indice de qualité introduit tout le long de ce chapitre n'est pas absolu.

# Chapitre 5

## Thermostat intelligent

### 5.1 Gestion automatique de la température et prévision des heures d'utilisation

Afin d'améliorer l'utilisation du thermostat intelligent, il a été nécessaire d'élaborer un algorithme permettant de déterminer les fréquences d'utilisation de ce dernier afin d'en anticiper son usage.

L'algorithme en question doit être capable de sauvegarder les données obtenues par les capteurs individuels. L'utilisation de fichiers a été choisie pour enregistrer et modifier ces informations. L'exploitation des bases de données ainsi constituées pourrait servir à la sauvegarde de telles informations, cependant le module sqlite3 nécessite quelques connaissances du langage SQL. De plus, les bases de données générées par sqlite3 sont destinées à l'enregistrement d'un grand nombre de données permettant de créer un tableau de bord sur le web.

Nous nous servons d'un dictionnaire dont les clés sont constituées de tuples. Les éléments de ces tuples reprennent le nom de l'arduino transmettant les informations et le jour de la semaine correspondant au jour de transmission de ces informations par l'arduino. Les valeurs associées à ces clés constituent des listes dont les éléments sont également des listes.

Pour concevoir ce programme d'intelligence artificielle, il a été nécessaire de séparer une journée en tranches de 10 minutes soit en 144 parties. En utilisant ces 144 tranches, des listes de 6 éléments sont générées. Dans l'ordre, ces éléments s'énumèrent comme suit : le numéro du jour de la semaine (allant de 0 à 6, le chiffre 0 correspondant au lundi), le numéro du jour de l'année, l'année, l'heure, le nombre de minutes, le nombre de passages de « False » à « True » en suite de la détection d'une présence et enfin le nombre de semaines écoulées reprenant l'analyse des habitudes de l'utilisateur. Il y a donc 144 listes rassemblées en une seule pour chaque arduino actif et ce pour chaque jour de la semaine.

On peut illustrer la création d'un tel fichier grâce à un tableau dont les colonnes représentent les arduino connectés et les rangées reprennent les jours de la semaine.

	arduino1	arduino2	arduino3	arduino4	arduino5
Lundi	liste1	...	...	...	liste5
Mardi	liste6	...	...	...	...
Mercredi	...	...	...	...	...
Jeudi	...	...	...	...	...
Vendredi	...	...	...	...	...
Samedi	...	...	...	...	liste30
Dimanche	liste31	...	...	...	liste35

Les listes dans chaque case sont construites suivant ce modèle :

jour de la semaine	jour de l'année	année	heure	minute	incréments	semaines écoulées
--------------------	-----------------	-------	-------	--------	------------	-------------------

Ici, le "nombre d'incrément" représente le nombre de fois que l'arduino a subitement détecté la présence d'un individu durant l'intervalle de temps défini par l'heure indiqué dans la liste et cette heure augmentée de 10 minutes.

La fonction permettant de savoir si l'on doit incrémenter la case "nombre d'incrément" se nomme "présence variation".

```

1 def presence_variation(sensor):
2     conn = sqlite3.connect("template.db")
3     cur = conn.cursor()
4     cur.execute("SELECT presence FROM sensor")
5     data = cur.fetchall()
6     cur.close()
7     presence = sensor.get_presence()
8     time.sleep(2)
9     if len(data)!=0:
10         if data[-1][0] == "false" and presence == "true":
11             res = "true"
12         else:
13             res = "false"
14     elif not data:
15         res="false"
16     return res

```

Cette fonction se repose sur la base de données utilisée par le tableau de contrôle. Durant l'exécution de cette fonction, l'objet "sensor" correspondant à un arduino transmet l'information d'une présence au serveur et compare cette information avec la donnée la plus récente de la base concernant également la présence d'un individu. Cet algorithme permet de déterminer l'heure d'arrivée d'un utilisateur et donc l'heure à laquelle l'information "présence" passe de "false" à "true". On incrémentera de 1 la case "nombre d'incrément" dans ce cas-ci.

La fonction responsable de l'incrémentation est décrite ci-dessous :

```
1 def increment(year,day_year,day,hour,min,sensor,string):
2     list=["Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"]
3     factor=int(min/10)
4     minute=factor*10
5     with open(directory+'presence_table.txt') as f:
6         monfichier=ast.literal_eval(f.read())
7         num=search_tuple(day,hour,minute,monfichier,string)
8         if presence_variation(sensor)=="true":
9             monfichier[string,list[day]][num][5] = monfichier[string,list[
10                day]][num][5] +1
11             if monfichier[string,list[day]][num][1]!=day_year:
12                 monfichier[string,list[day]][num][6] = monfichier[string,list[
13                   day]][num][6]+1
14             monfichier[string,list[day]][num][1]=day_year
15             with open(directory+'presence_table.txt','w') as f2:
16                 f2.write(str(monfichier))
```

Il est à noté que pour obtenir les arguments tel que l'année, le jour de l'année, le jour de la semaine et l'heure il a été nécessaire d'utiliser le module "time" de python. On utilise l'algorithme "présence variation" afin de incrémenter la case "nombre d'incrément". Ceci est réalisé seulement si "présence variation" renvoi "true".

On remarque ici que la composante "jour de l'année" de chaque élément de liste permet de déterminer le nombre de semaines écoulées depuis le début de l'étude des habitudes de l'utilisateur.

La fonction "search tuple" fait correspondre le jour et l'heure actuelle avec un élément d'une liste qui représente la valeur de la clé ("Arduino","Jour de la semaine").

Il reste donc à décrire le code responsable de la gestion automatique de la température. Une commande réactive est incorporée afin que la régulation de température s'enclenche lors de la détection de présence d'un individu.

```
1 if presence=="true":
2     value=pid_regulation(sensor, 25)
3     sensor.set_valve(value)
4     time.sleep(2)
```

Selon la période d'analyse des habitudes d'utilisation du thermostat, la commande prévoyant les heures d'utilisation permet de réguler la température de deux façons :

- si la période d'analyse n'est pas atteinte, on utilise la fonction "increment" pour modifier le tableau de données.
- si la période d'analyse est atteinte, on sélectionne dans les éléments de la liste ceux qui ont le plus grand nombre d'incrément. Ces éléments sont trouvés en triant par sélection la liste (cette liste étant la valeur de la clé "arduino" et "jour de la semaine"). Le groupe a

décidé de prendre uniquement trois éléments de ces listes par jour et par arduino enregistré.

Dans les deux cas présentés, l'absence d'un individu durant une durée de quinze minutes désactive la vanne de chauffage. L'algorithme constituant l'intelligence artificielle sera utilisé dans la fonction "main" qui elle est exécutée dans une boucle infinie.

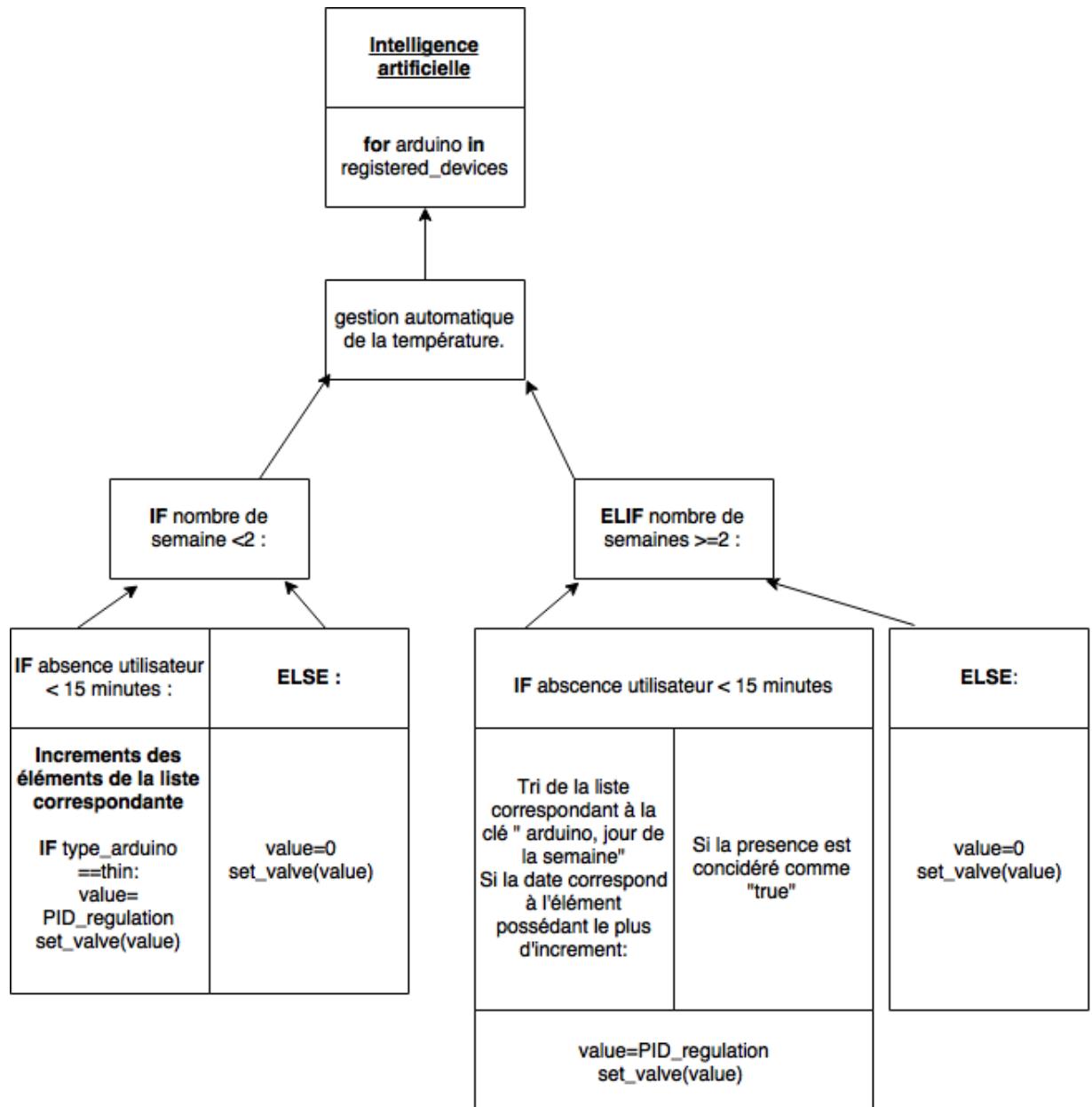


FIGURE 5.1 – Diagramme de l'algorithme composant l'intelligence artificielle

## 5.2 Interface d'utilisateur

Dans cette section, nous allons nous consacrer aux différentes fonctionnalités et interactions possibles pour le contrôle du thermostat par l'utilisateur. Pour commencer, une page web est à la disposition de l'utilisateur via l'adresse suivante : `http://192.168.10.1:8080`. Il est nécessaire de se connecter au réseau WiFi du thermostat central pour accéder à cette page ou que celui-ci est connecté au réseau domestique. Il pourra s'y connecter via n'importe quel navigateur internet et ainsi donc voir la température de chaque pièce dans laquelle se trouve un thermostat individuel. En effet, cette page met à jour la température régulièrement en se basant sur les dernières mesures enregistrées par l'unité centrale dans la base de données. Grâce à cette même page, l'utilisateur sera en mesure de modifier la température de consigne de son bâtiment en fonction de ses besoins. Une autre option permet également d'arrêter le processus de chauffe en cas de nécessité.

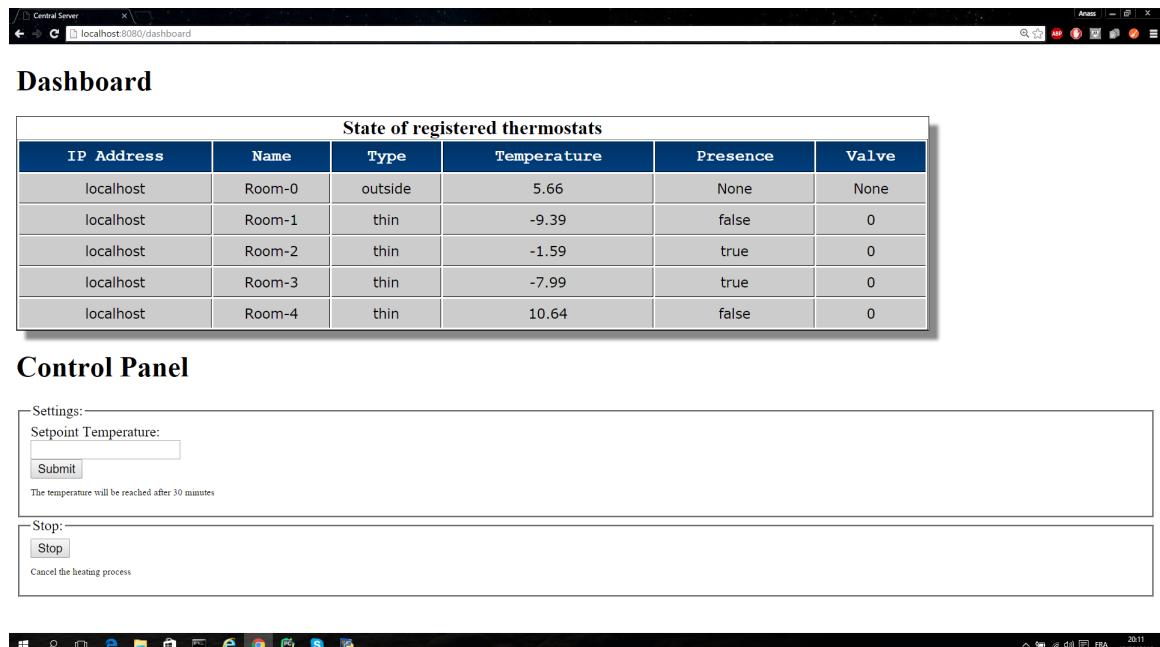


FIGURE 5.2 – Dashboard

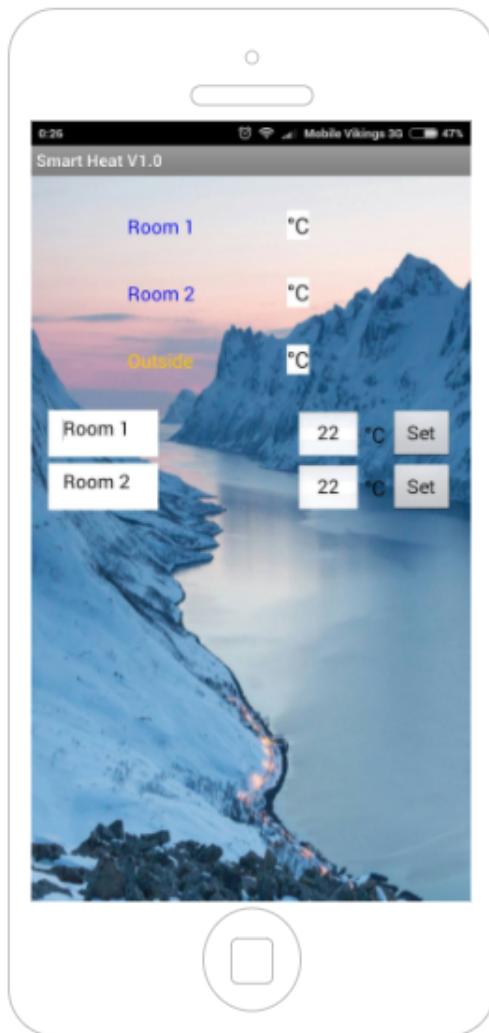


FIGURE 5.3 – Application android

En parallèle, nous avons également mis au point une application Android (figure 5.3) simple et intuitive qui permet à l'utilisateur d'avoir un contrôle de son thermostat à partir de son smartphone. Son développement a été possible grâce au logiciel App Inventor 2 [3] qui est un programme développé par Google et entretenu par le Massachusetts Institute of Technology (MIT). Cette application permet, tout comme la dashboard, de modifier la consigne et afficher la température des différentes pièces de l'immeuble.

# Chapitre 6

## Fonctionnement du groupe

Cette section du rapport est dédié au fonctionnement du groupe afin d'analyser les points forts et points faibles du groupe responsable de la conception du thermostat et de décrire au mieux son organisation.

La conception d'un thermostat intelligent est un projet nécessitant l'apprentissage du travail en équipe. Cet apprentissage est important pour tout ingénieur ou scientifique qui est amené à collaborer avec plusieurs personnes. Il a donc fallu mettre en place une méthode de travail où la communication est un élément clef notamment pour la répartition des tâches.

La communication entre les membres du groupe a été assurée par la mise en place de plusieurs réunions organisées chaque semaine. Lors de celles-ci deux membres sont désignés pour être soit animateur soit secrétaire. L'animateur a pour rôle de donner la parole et de faire participer tout le monde aux discussions, le secrétaire quant à lui reprend par écrit les grandes lignes de la réunion. La rédaction de P.V (procès verbal) permettent de rester informé sur les taches à accomplir aux prochaines réunions ainsi que du lieu et de la date ou se tiendront celles-ci.

Il a été préférable de créer un groupe sur le logiciel GIT, un logiciel de gestion de fichier conservant un ensemble de fichiers ainsi que les modifications récentes effectuées sur ceux-ci. Ce logiciel a permis d'enregistrer chaque procès-verbal rédigé lors des réunions et ainsi donc dévoiler une certaine évolution dans l'avancement du projet. Cependant, sur la figure 6.1, nous pouvons facilement remarquer que l'utilisation du GIT était assez faible durant le premier quadrimestre, le groupe n'était pas en mesure d'utiliser pleinement les ressources fournies. Néanmoins, nous pouvons remarquer une augmentation du nombre d'utilisation du GIT. En effet, les échelles des deux graphiques (figure 6.1) sont différentes. Le diagramme "b" du GIT possède une échelle allant au-delà de 10 uploads par jour.

Les références des ouvrages utilisés ont été réunis grâce au logiciel « Zotero » permettant à chaque membre de se renseigner sur un sujet à partir d'un ouvrage fourni par un autre étudiant.

L'entente des membres du groupe s'est vu améliorée étant donné que ceux-ci appartiennent au groupe 3B de TP et on donc un horaire commun. En raison du nombre de debuggage à effectuer et au manque de temps du fait de la densité des cours de l'année BA2, plusieurs réunions sans le tuteur ont été planifiées pour se tenir informé de l'avancement ds travaux des différents membres de l'équipe. Ces réunions avaient pour but d'éclaircir toute incompréhension

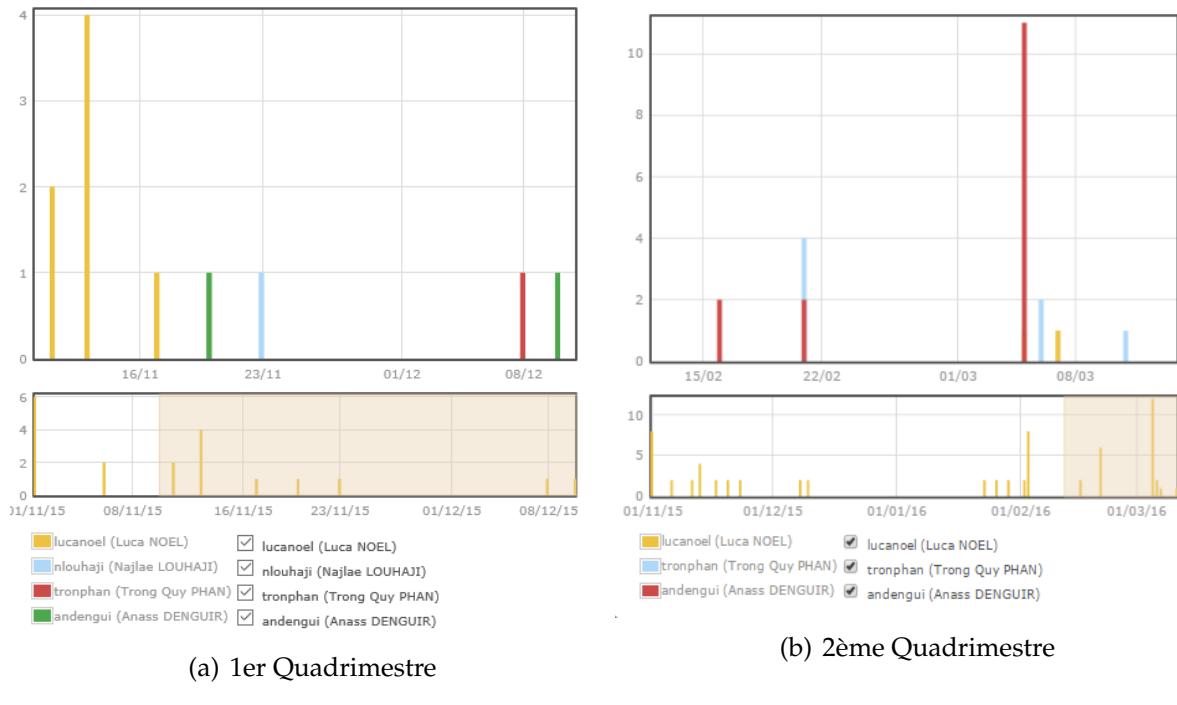


FIGURE 6.1 – Diagrammes du GIT

afin que chaque membre de l'équipe puisse être imprégné de l'état d'avancement du projet.

L'équipe a connu des moments difficiles à gérer. En effet, suite à quelques malentendus, certains problèmes concernant la conception ou les échéances ont été rencontrés. Heureusement, ces problèmes ont généralement pu être réglés ou contournés grâce à une solution alternative.

Pour conclure, un plan de travail a été mis en place lors du premier quadrimestre. Celui-ci a permis au groupe de se projeter sur l'année afin d'avoir une idée globale des différents volets touchant le projet. Ce plan a également servi à fixer des échéances que l'équipe a tenté de respecter. Durant le second quadrimestre, nous nous sommes concentrés sur la partie intelligente du projet ainsi que sur la régulation PID du thermostat. On remarque donc que le planning est respecté. Les tâches citées précédemment constituaient la principale préoccupation du groupe sans compter les problèmes de communication entre Raspberry et arduino, le plan de travail n'a donc pas pu être mis à jour.

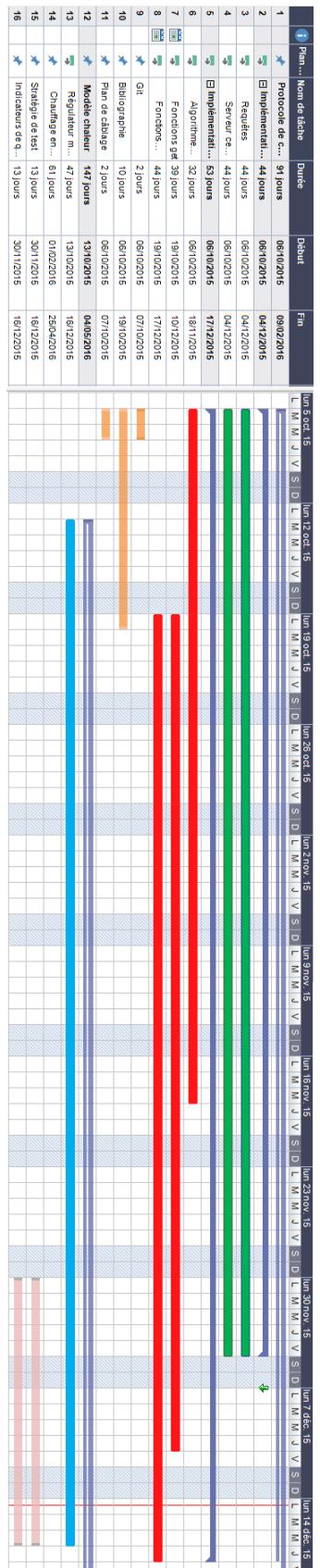


FIGURE 6.2 – Plan de travail

# Chapitre 7

## Conclusion

Arrivé au terme de ce projet, ce rapport reflète les 6 mois de travail de groupe et son déroulement. Un des plus gros défis dans ce projet a été l'implémentation de l'algorithme pour la régulation de la température en une période de temps très courte. Le choix se portant sur une régulation PID a été très laborieux par le développement d'une phase intermédiaire de calibration afin de trouver des constantes adéquates. En effet, une des grosses difficultés de ce projet résidait dans le fait que les environnements dont la température devait être contrôlée ne possédaient pas des caractéristiques physiques fixes et une loi de comportement thermique connue. Le thermostat intelligent devait donc s'adapter à tout type de pièce. Cette contrainte nous a empêché de développer une simulation informatique de notre environnement dans le but de réaliser nos tests de calibration et de régulation rapidement et non pas sur l'environnement physique de simulation pour lequel un temps de refroidissement était nécessaire entre chaque test.

Une autre difficulté et frustration de ce projet a été le débogage du code python lié aux différents problèmes de communication entre les thermostat central et individuels. Ces bugs difficiles à régler étaient assez récurrents et contraignants notamment lorsqu'il était nécessaire de réguler la température durant une journée ou tester notre intelligence artificielle sur une longue période de temps.

Ce projet multidisciplinaire touchant à sa fin, nous pouvons tirer une leçon sur les différentes erreurs commises durant l'année en particulier lors du premier quadrimestre. La gestion du temps imposé a probablement été notre plus grand point faible ainsi que notre négligence lors de la présentation de mi-parcours.

Malgré tout, cela fut une expérience très enrichissante nous montrant d'autres aspects dans le domaine de l'informatique, l'électronique, la domotique et la gestion de projet. Ce travail nous a également permis de développer notre esprit d'ingénieur, en effet un ingénieur se doit de trouver une solution optimale à son problème et le plus rapidement possible.

Enfin ce projet n'étant pas totalement optimisé, des améliorations et innovations sont parfaitement envisageables tant bien au niveau de la communication, de l'intelligence artificielle ou encore de l'interface d'utilisateur.

# Bibliographie

- [1] Arduino - PWM. Site web sur internet. <<https://www.arduino.cc/en/Tutorial/PWM>>. Consulté le 7 décembre 2015.
- [2] Commande numérique de systèmes dynamiques : cours d'automatique - Roland Longchamp - Google Livres. Site web sur internet. <<https://books.google.be/books>>. Consulté le 1 mars 2016.
- [3] MIT App Inventor | Explore MIT App Inventor. Site web sur internet. <<http://appinventor.mit.edu/explore/>>. Consulté le 13 mars 2016.
- [4] Apache2. *Requests : HTTP for Humans*. Site web sur internet. <<http://www.python-requests.org/en/latest/>>. Consulté le 11 novembre 2015.
- [5] Arduino. *Arduino projects book*. Project and text by Scott Fitzgerald and Micheal Shiloh, 2013.
- [6] Bottle. *Bottle : Python Web Framework* . Site web sur internet. <<http://bottlepy.org/docs/dev/index.html>>. Consulté le 11 novembre 2015.
- [7] Jean-François BOURGEOIS. *Automatisme et régulation des équipements thermiques. Techniques de l'ingénieur Fluides, contrôle et isolation thermiques*, base documentaire : TIB595DUO.(ref. article : be9590), 2015. fre.
- [8] World Wide Web consortium. *HTML5 A vocabulary and associated APIs for HTML and XHTML*. site web sur internet. <<http://www.w3.org/TR/html5/>>. Consulté le 11 novembre 2015.
- [9] DE GEEST Emmanuel. *Méthodes d'optimisation pour le réglages de contrôleurs PID*. Pdf sur internet.<<http://www.montefiore.ulg.ac.be/systems/degeest.pdf>>. Consulté le 12 novembre 2015.
- [10] HAELETERMAN Marc. *Physique générale : Université libre de Bruxelles*. BA1 - Syllabus de Laboratoire (2014-2015). ULB, 1e édition-tirage edition, 2014.
- [11] National Instruments. *Types de capteurs de température - National Instruments*. Site web sur internet. <<http://www.ni.com/white-paper/10635/fr/>>. Constulté le 11 novembre 2015.
- [12] JSON. *JSON*. Site web sur internet.<<http://www.json.org/>>. Consulté le 11 novembre 2015.
- [13] MCROBERTS Michael. *Beginning Arduino*. Technology in action. Apress, October 2013.
- [14] Raspberry Pi. *Raspberry Pi - Teach, Learn, and Make with Raspberry Pi*. Site web sur internet. <<https://www.raspberrypi.org/>>. consulté le 19 octobre 2015.
- [15] Python. *sqlite3 interface for SQLite databases*. Site web sur internet. <<https://docs.python.org/2/library/sqlite3.html>>. Consulté le 11 novelbre 2015. url = <https://docs.python.org/2/library/sqlite3.html>, urldate = 2015-11-11.

- [16] SWIMMEN Gérard. *Apprendre à programmer avec Python 3*. Noire. Eyrolles, 2012.
- [17] Vishay. *NTC Thermistor, radial leaded, standard precision*. PDF sur internet.<<http://www.vishay.com/docs/29049/ntcle100.pdf>>. Consulté le 11 novembre 2015.
- [18] Wikipedia. *Capteur*, November 2015. Site web sur internet. <<https://fr.wikipedia.org/wiki/Capteur>>. Consulté le 11 novembre 2015. Page Version ID : 120198735.

# Annexe A

## Rapport de recherche bibliographique

La réalisation d'un thermostat intelligent nécessite des renseignements sur l'aspect informatique du projet comprenant, par exemple, les algorithmes de contrôle ainsi que les constructions de circuits électriques à produire.

Une recherche bibliographique semble donc indispensable . Les équations de recherche ont permis de trouver les sources que les membres du groupes ont décidé de sélectionner. Ces équations ont été créé en se basant sur les mots-clé provenant du cahier des charges. Les sources sélectionnés sont accompagnés , dans ce document, des motivations de sélection. Ce document se conclut par la bibliographie. Celle-ci contient chaque détails de sources choisis.

### A.1 Ressources consultées

Bureau d'appui pédagogique de l'école polytechnique (BAPP), Google, Wikipédia, catalogue de l'ULB (CIBLE +), Technique de l'ingénieur.

### A.2 Mots-clés

#### A.2.1 Keywords

Thermostat, sensor, actuator, Raspberry Pi, Arduino, Wi-Fi module (ESP8266), thermal inertia, thermistor, power resistor, hypertext transfert protocol, server, python, communication protocol, database,implementation, controller algorithm, on-off.

#### A.2.2 Traduction des mots-clés

Thermostat, capteur, actionneur, Raspberry Pi, Arduino, module Wi-Fi(ESP8266), inertie thermique, thermistor, résistance de puissance, hypertext transfert protocol, serveur, python, protocole de communication, base de données,implémentation, algorithme de contrôle, on-off.

Mots-clés.	Keywords.
Thermostat individuel	Thin thermostat
Capteur	Sensor
Actionneur	Actuator
Raspberry Pi	Raspberry Pi
Arduino	Arduino
Module Wi-Fi (ESP8266)	Wi-Fi module (ESP8266)
Inertie thermique	Thermal inertia
Thermistor	Thermistor
Resistance de puissance	Power resistor
World Wide Web	World Wide Web
Thermostat central	Central thermostat
Hypertext transfert protocol	Hypertext transfert protocol
Serveur	Server
Python	Python
Protocol de communication	Communication protocol
Base de données	Database
Implémentation	Implementation
Algorithme de contrôle	Controller algorithm
On-Off	On-Off

### A.3 Sources sélectionnées

**The Raspberry Pi Foundation.** *Help Guides and resources - how to use raspberry-pi.* [14]

Lors de la réalisatson du thermostat intelligent, la compréhension du fonctionnement du Raspberry Pi sera nécessaire. Ce document nous permettra de nous renseigner sur la manière efficace d'exploiter l'appareil. La documentation est fournie par les développeurs du Raspberry PI. La validité de la source est donc vérifiée.

**National instruments.** *Types de capteurs de température.* [11]

**Wikipédia,** *Capteur* [18]

L'utilisation de capteur thermique ou thermistor fera l'objet de plusieurs recherches. Le groupe devra notamment rechercher le type de capteurs auquel il fait face, se renseigner sur son fonctionnement ainsi que sur l'exploitation des données fournies par le capteur.

La validité d'un site tel que Wikipédia n'est pas assurée. Néanmoins, le contenu des documents trouvés sur la page web est fort similaire aux documents fournis par National instruments. On peut donc juger la source de Wikipédia comme valide.

**MICROBERTS Michael.** *Beginning Arduino.* [13]

**FITZGERALD Scott.** Mai 2013. *Arduino project book.*[5]

Le Raspberry Pi utilisé durant le projet sera relié à un ou plusieurs thermostats individuels. Une carte Arduino permettra à un thermostat de réaliser ses fonctions ainsi que d'envoyer des

données au Raspberry PI considéré ici comme un thermostat central. L'étude de son fonctionnement ainsi que de son langage de programmation seront donc nécessaires durant la réalisation d'un thermostat individuel. On peut considérer la source comme valide. En effet, il s'agit du livre écrit par l'organisme Arduino afin de donner des connaissances de base aux utilisateurs.

#### **Haelterman Marc. 2014. *Physique général : Syllabus de laboratoire*. [10]**

La création de circuit électrique fera partie de la conception du thermostat intelligent. Les informations du cours de physique général seront donc utiles à la construction. Le syllabus de laboratoire comprend la description de deux manipulations de circuit électriques. Ce syllabus était proposé aux étudiants par le professeur Marc Haelterman durant l'année 2014-2015. On peut considérer le document pertinent.

#### **Requests : HTTP for Humans [4]**

La librairie "request", inclue dans le programme python, servira à la communication entre les thermostats individuels et le Raspberry Pi. La compréhension du package nous permettra de réaliser la distribution de données entre les deux composantes. Le site est proposé par le tuteur du groupe ainsi que par le cahier des charges. Il contient des exemples de réalisation de programme et des explications permettant un apprentissage rapide et efficace. Cette source est proposée par les rédacteurs de l'énoncé du projet. On conclut donc la pertinence du site.

#### **HTML5 A vocabulary and associated APIs for HTML and XHTML [8]**

Lors des échanges de données entre le Raspberry Pi et les thermostats individuels, le Raspberry servira de serveur. Les thermostats individuels enverront donc leurs requêtes au serveur. Il est donc nécessaire de pouvoir gérer des pages html. Le site consulté est proposé et créé par le World Wide Web Consortium qui est une communauté internationale. Cela permet de vérifier la validité du document. Il traite le sujet du HTML et permet de sélectionner des informations intéressantes bien que, dans le cadre de ce projet, la gestion du serveur se fera via python.

#### **Apprendre à programmer avec Python 3.[16]**

Le document de Gérard Swimmen servira de rappel mais renseignera également les membres du groupe sur les modules inclus dans python. Il sera également d'une grande aide en ce qui concerne la conception d'un serveur. Il s'agit d'un document didactique conseillé par Thierry Massart, professeur d'informatique à l'université libre de Bruxelles.

#### **Bottle : Python Web Framework [6]**

Bottle est un module python qui permet en seulement quelques commandes d'implémenter un serveur. Il comporte également quelques méthodes utiles telles que GET et POST qui permettent d'interagir avec ce dernier. L'utilisation de ce module ainsi que les sources liées à celui-ci a été mentionnée dans le cahier des charges. Ce site est proposé par les rédacteurs de l'énoncé du projet.

#### **SQLlite3 interface for SQLite databases[15]**

SQLite3 est une librairie Python qui sert à l'implémentation de base de données. Ces dernières serviront notamment à gérer les diverses informations communiquées par les capteurs. SQLite est simple d'utilisation et déjà installé par défaut sur python, ce qui facilite son utilisation. La pertinence du document est certifiée car l'organisme python est à l'origine de celui-ci.

### **JSON [12]**

Json est un texte formaté qui sert à la communication entre le serveur et les différents thermostats individuels. Ce langage est particulièrement utile car il présente une structure similaire à celle des langages de programmation tel que python. Ce site a été conseillé dans le cahier des charges.

### ***Automatisme et régulation des équipements thermiques* [7]**

#### ***Méthodes d'optimisation pour le réglages de contrôleurs PID* [9]**

Il est souhaité que, lorsque l'un des thermostat individuel chauffe l'environnement physique, nous n'observions pas une oscillation de la température au niveau de la température de consigne. Il faut donc créer un algorithme de contrôle basé sur certaines modélisations. "Automatisme et régulation des équipements thermiques" nous renseigne sur le fonctionnement du PID. Il s'agit d'un document provenant du site "techniques ingénieur" connu pour ses sources fiables. "Méthodes d'optimisation pour le réglages de contrôleurs PID" nous donne chaque détails concernant le PID. Ce document est un mémoire d'un ingénieur diplômé de l'université de Liège.

### ***NTC Thermistor, radial leaded, standard precision* [17]**

Le calcul de la température par le thermostat individuel sera effectué grâce à un thermistor. Le fichier PDF est fourni par Vishay.com, un site spécialisé dans la vente de matériels électronique. Le document donne des renseignements tel que l'algorithme de conversion de bit en température ce qui sera utile à la conception du thermostat.

## **A.4 Equations de recherches**

*Help Guides and ressources - how to use raspberry-pi. :*

Raspberry pi AND Tutorial

*Types de capteurs de température. :*

Sensor AND categories

*Généralités sur les capteurs. :*

Sensor AND categories

*Capteur :*

Sensor

*NTC Thermistor, radial leaded, standard precision :*

thermostat AND sensor AND Arduino

*Automatisme et régulation des équipements thermiques :*

thermal inertia AND regulation

*HTML5 A vocabulary and associated APIs for HTML and XHTML :*  
HTML AND World Wide Web

*Requests : HTTP for Humans :*  
Python AND Communication protocol

*Bottle : Python Web Framework :*  
Python AND Server AND World Wide Web

*SQLite3 interface for SQLite databases :*  
Python AND Database

*JSON :*  
Python AND Communication protocol

*Automatisme et régulation des équipements thermiques :*  
Implementation AND thermostat AND Controller algorithm

*Méthodes d'optimisation pour le réglages de contrôleurs PID :*  
Implementation AND Controller algorithm AND On-Off

## Annexe B

### Information sur le groupe



DENGUIR Anass  
000408788  
[andengui@ulb.ac.be](mailto:andengui@ulb.ac.be)



LOUHAJI Najlae  
000408719  
[Najlae.Louhaji@ulb.ac.be](mailto:Najlae.Louhaji@ulb.ac.be)



NOEL Luca  
000407261  
[Luca.Noel@ulb.ac.be](mailto:Luca.Noel@ulb.ac.be)



PHAN Trong Quy  
000413033  
[tronphan@ulb.ac.be](mailto:tronphan@ulb.ac.be)