Rapport Final PFE

Création d'un banc d'essai dans la perspective d'amélioration des régulations thermiques des salles de l'école

Florian Royer & Zohour Assaieb

24 févr. 15

Remerciement à notre tuteur école Mr Belkacem Ould Bouamama pour son aide précieuse et sa capacité de suivi de notre projet. Merci aussi à l'équipe Wago, plus précisément à M. Loïc Fricou qui a su apporter une assistance technique rapide lors des différentes difficultés rencontrées.

PREAMBULE

Ce rapport de fin de projet d'étude résume le travail effectué dans le cadre de notre formation en dernière année à Polytech Lille, département Informatique Micro-électronique et Automatique. Le projet s'inscrit dans le contexte d'économie d'énergie et de l'effort en matière de développement durable que fournit l'école. Le but du projet étant de mettre en place un banc de test afin de réguler la température d'une salle, puis une fois le banc de test en place, procéder à des phases de tests afin d'améliorer la régulation thermique.

Le rapport se découpe en 3 grandes parties qui correspondent chacune à une étape de développement du projet :

- Partie théorique : étude des normes utilisées à Polytech, de l'installation de la robinetterie et modélisation d'un radiateur et de son environnement sous Matlab ;
- Partie commerciale : étude de différents composants qui seront à utiliser, appels d'offre auprès des entreprises et passage de commande ;
- Partie mise en place du banc de test : Montage des différents composant du système et création d'un programme pour l'automate, création d'une interface graphique permettant de récupérer des données sur l'automate, confrontation théorie-Pratique, estimation du rendement.

TABLE DES MATIERES

1 PARTIE THEORIE

1.1 CAHIER DES CHARGES

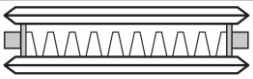
Cette partie du projet s'est principalement déroulée de septembre à début janvier. Elle consistait à répondre à un cahier des charges que nous avons fixé au départ :

- Etude bibliographique afin de comprendre les échanges thermiques et les différentes normes utilisées ;
- Modéliser le système Radiateur et son environnement :
  - Etablir le modèle mathématique ;
  - Les échanges internes ⇔ BondGraph ;
  - Le modèle fonctionnel sur Matlab-Simulink ;
- Etablir différentes commandes (Floue et PID) et déterminer la plus pertinente ;
- Discrétiser le système (commande + modèle) afin de se rapprocher de la réalité.

1.2 ETUDE BIBLIOGRAPHIQUE ET DE L'EXISTANT

1.2.1 Dimensionnement du radiateur de test

A l'heure actuelle, les salles de Polytech sont composées de radiateurs à eau chaudes réglés par vanne thermostatique. Ce type de vanne permet de fixer une température avec une très faible précision. Dans le cadre de notre projet et pour pouvoir ensuite modéliser un radiateur, nous avons fixé le type de radiateur suivant :

Matériaux	**Illustration**	**Dimensions**	**Puissance P50 (W/m²)**	**Pente d'émission**
Panneaux en acier horizontaux		- Hauteur de 0.7m - Longueur de 2m - Profondeur de 10 cm	2 220	1.33

La vanne thermostatique utilisée dans la plupart des salles permet de choisir un mode de confort entre 0 et 6, mais ne permet en aucun cas le contrôle de la température. Après le mode de confort choisi, la vanne utilise le principe naturel de la dilatation de la matière à l'état gaz (1 : bulle de gaz) pour ensuite décompresser/compresser le ressort (4) qui lui réglera le débit avec le piston (5).

Ce type de vanne provoque une très grande incertitude (~3°C) sur la température du radiateur car le débit dépendra de la température et de la pression.

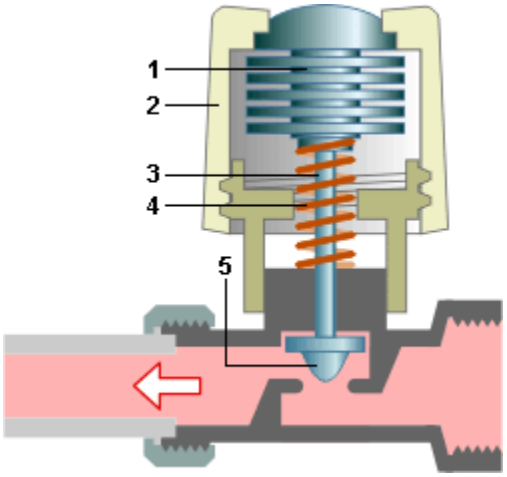


Figure 1 : illustration d'une vanne thermostatique

Le réseau de radiateur est de type parallèle bitube. L'eau chaude arrive de la chaudière, et l'eau de sortie du radiateur retourne directement à la chaudière.

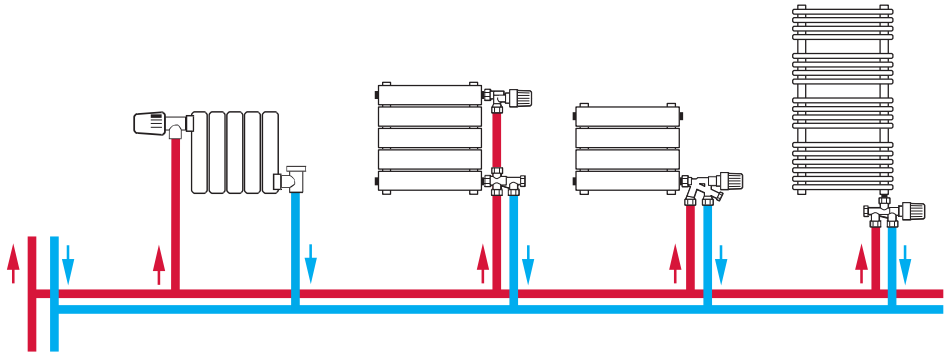


Figure 2 : Schéma simplifié d'une installation bitube

Il existe d'autres réseaux qui ne sont pas considérés dans le cadre de notre projet puisqu'ils ne représentent pas le réseau hydraulique de Polytech.

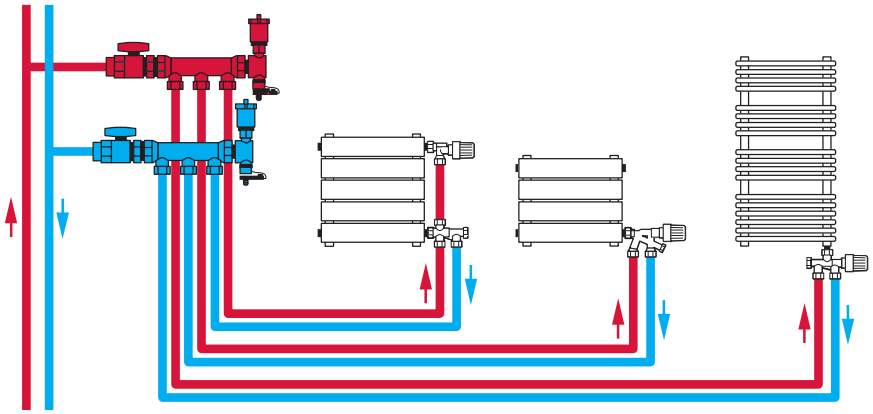


Figure 3 : Schéma simplifié d'une installation hydro câblée

1.2.2 Les échanges de chaleur

Afin de pouvoir faire une modélisation correcte du radiateur de test, il est aussi important de définir les échanges thermiques mis en jeu. L'échange de chaleur se fait principalement par :

- Convection : transfert de chaleur par la matière, d'atome vers atome. Ce transfert implique un déplacement de la matière. La valeur de la puissance émise par convection dépend principalement de la hauteur de la pièce et de la hauteur à laquelle est placé le radiateur, ainsi que la surface du radiateur.

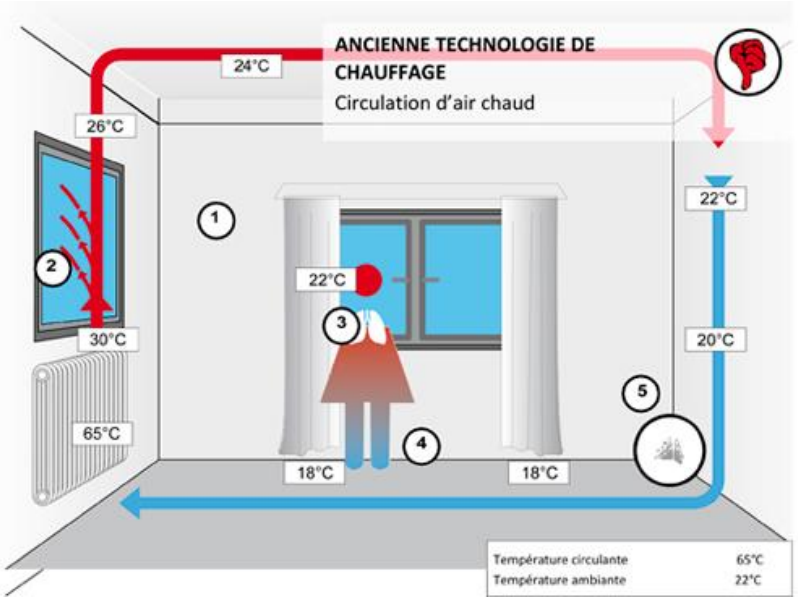


Figure 4 : Illustration de la convection

- Rayonnement : une partie du transfert se fait aussi par rayonnement électromagnétique. Quelle que soit sa température, un corps émet un rayonnement thermique, celui-ci est plus ou moins intense selon la température du corps. La longueur d'onde à laquelle est émise ce rayonnement dépend aussi de cette température.
- Conduction : ce phénomène est très peu mis en jeu lors de l'échange radiateur vers salle car l'air est très peu conducteur thermique. Par contre, c'est le phénomène thermique qui est le plus mis en jeu lors du passage de l'eau chaude dans le radiateur.

1.2.3 Norme de température utilisée

Il existe plusieurs normes concernant la température de chute dans un radiateur. Celle-ci correspond à la moyenne entre température d'entrée et température, en retirant la température ambiance normalisée de 20°C.

- Avant 1997, NF P52011 : $\Delta T = \frac{(90+70)}{2} - 20 = 60°C$ (Norme Française)
- Après 1997, EN 442 : $\Delta T = \frac{(75+65)}{2} - 20 = 50°C$ (Norme Européenne)

Dans le cadre du projet, on considère la Norme Française d'avant 1997, car Polytech dépend de la chaudière de l'université qui date d'avant 1997.

1.3 ETABLISSEMENT DU MODELE

1.3.1 Schématisation 'électronique'

Afin de mieux comprendre les pertes et capacités mis en jeu, et du peu de connaissances en Thermodynamique et mécanique des fluides, nous avons débuté par un schéma électrique.

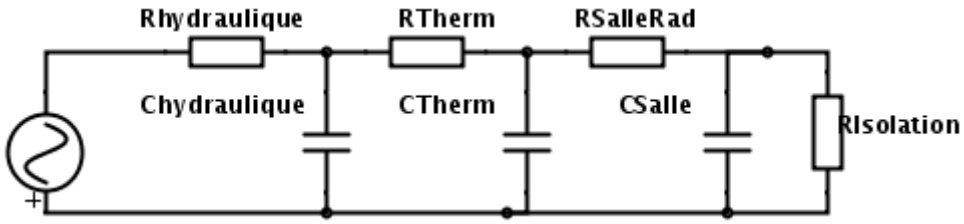


Figure 5 : Représentation électrique du système

La Source de tension correspond au flux d'enthalpie fourni à une température donnée et un débit d'eau chaude fixé. Voici les différents éléments du schéma :

- $R_{hydraulique}$: Résistance à l'entrée du radiateur provoquant une faible perte de pression
- $C_{hydraulique}$: correspond à la capacité hydraulique du radiateur
- $T_{hermique}$: Lors de l'échange entre l'eau et le radiateur par conduction, il y a une perte thermique
- $T_{hermique}$: Capacité du radiateur à stocker une température
- $R_{Salle/radiateur}$: pertes thermique dues aux matériaux du radiateur, de sa longueur d'onde de rayonnement, et des pertes lors de l'échange par convection
- C_{salle} : Correspond au produit de la capacité thermique de l'air avec le volume de la salle.

1.3.2 Bond Graph et schéma bloc

Le bond graph est un outil plus adéquat pour représenter le système car différentes forme d'énergies sont en jeu (hydraulique, thermique, mécanique).

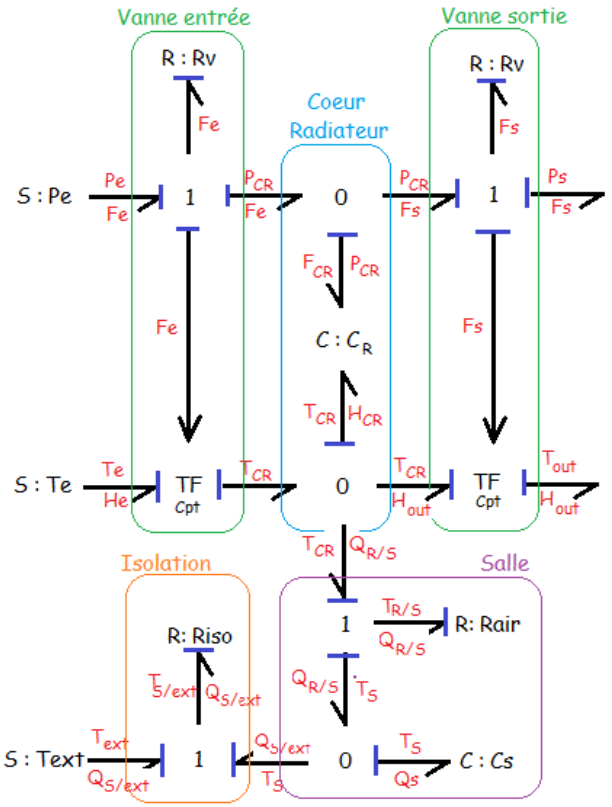


Figure 6 : Bond Graph du système

Cette représentation nous permet ensuite de créer le schéma bloc du système (disponibles sur le Wiki).

1.3.3 Modélisation du radiateur

Afin de comprendre le fonctionnement exact du radiateur il a fallu se documenter. On a alors déduit les formules de capacité hydraulique du radiateur :

$$Q_{radiateur} = D.C_{p\,eau}.\Delta T$$

Et nous avons utilisé :

$$T_m = \frac{1}{C_{p\,eau}} \int (H_{entré} - Q_{Radiateur \rightarrow salle} - H_{sortie})\, dt$$

- H entant l'enthalpie thermique
- $C_{p\,eau}$ capacité thermique de l'eau à pression constante.

1.3.4 Modélisation de l'échange radiateur - salle

$$\begin{cases} Q_{fournie} = Q_{Nominale}\left(\frac{\Delta T}{60}\right)^n \; avec\; \Delta T = \frac{T_d + T_r}{2} - T_s \\ Q_{Radiateur \rightarrow salle} = Q_{Rayonnement} + Q_{Convection} = \Delta T.S.(h_c + h_r) \\ Q_{fournie} = Q_{Radiateur \rightarrow salle} + Q_{Pertes} \end{cases}$$

- T_d : Température de départ dans le tuyau 90°C dans le cas de Polytech

- T_r : Température de retour, eau froide retournant à la chaudière (Varie suivant le débit d'eau chaude d'entrée)
- Q_N : Puissance estimée du radiateur (en W) à la norme NF $\Delta T_{Nominale}$=60°C
- H_c : coefficient de transmission thermique par convection
    - $hc = 5{,}6.(\frac{\Delta T}{T_s.h})^{0.25}$
    - h étant la position en hauteur du radiateur (on estime 0.7 m)
- h_r : coefficient de transmission thermique par rayonnement
    - $h_r = Ec\ x\ (T_m + T_s)\ x\ (T_m^2 + T_s^2)\ x\ C$
    - T_m étant la température moyenne du radiateur
    - T_s la température du local
    - C : constante de Stefan-Boltzmann, et vaut $5{,}67051\ .10^{-8}$
- S : la surface du radiateur, surface d'échange dimensionnée à 2,5 m²
- D_{eau} : Débit d'eau d'entrée

Une fois la modélisation aboutie, nous nous sommes focalisés sur la commande du modèle.

1.4 ETABLISSEMENT DE LA CONSIGNE

Nous avons développé différentes consignes :

- PID continu puis PID discret ;
- Commande floue.

1.4.1 Commande par PID

Le bloc PID développé prend en entrée la différence de la température mesurée et la température désirée, et en sortie une valeur de 0 à 100 % de consigne d'ouverture de la vanne. L'outil ''Tune'' de Matlab nous a ensuite permis d'obtenir les valeurs du PID adéquates (Kp, Ki et Kd). On obtient alors les valeurs ci-dessous (à noter que le système est discrétisé) :

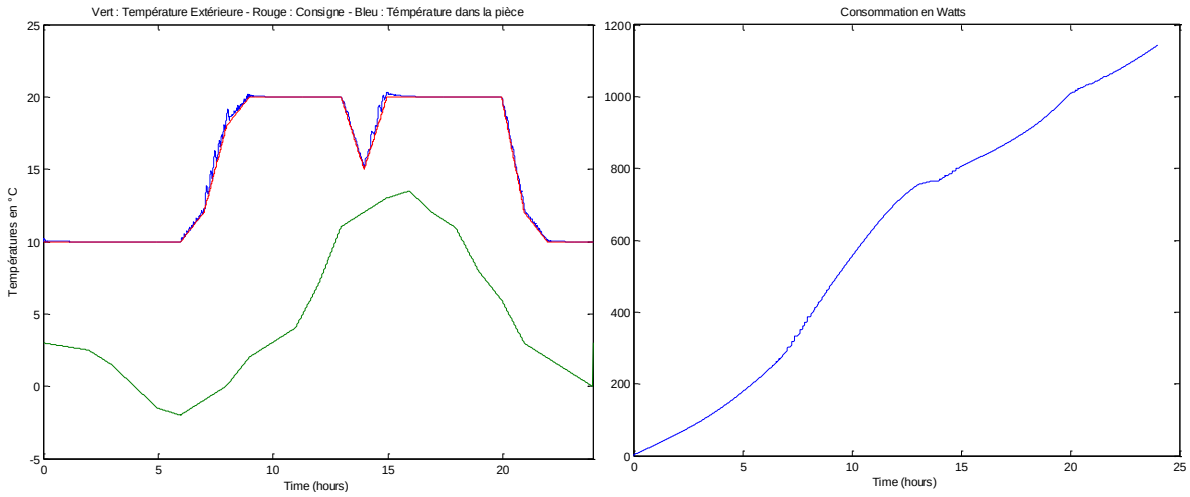


Figure 7: Aperçu de la régulation par PID et Consommation à droite

Conditions de test : La surface de la salle est de 50 m² pour une hauteur de 2.5m, ce qui représente une grande salle comme la A312 ou les salles pouvant contenir jusque 60 personnes.

On s'aperçoit que le PID calculé permet d'obtenir une grande robustesse du système, mais une erreur assez importante d'environ ~0.5°C. La consommation estimée in fine est de 1150 Watts/jour.

Malheureusement, nous avons une légère erreur de modélisation que nous n'arrivons pas à corriger : Le modèle ne prend pas en compte le retard du radiateur, le déchargement de capacité thermique qu'il a atteint, d'où la chute brutale de température dans la pièce lorsque la consigne diminue.

1.4.2 Commande floue

La commande floue est utilisée lorsque les équations différentielles de la modélisation mathématique du système ne sont pas linéaires ou lorsqu'il est difficile de modéliser un système. Comme nous avons une difficulté à modéliser le retard du radiateur, nous avons décidé de tester notre système avec une commande à logique floue. Pour se faire, nous avons utilisé la différence de température entre celle désirée et la mesurée, ainsi que la dérivée de cette différence pour savoir si la pièce se réchauffe ou se refroidit.

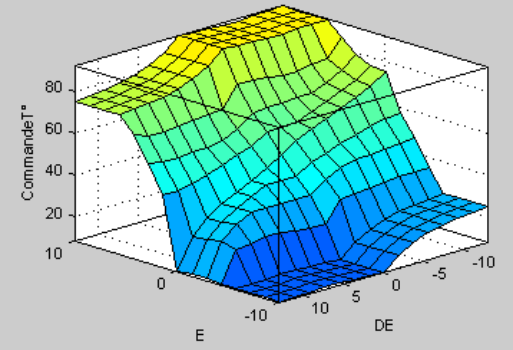


Figure 8: Surface Viewer FuzzyLogic

La Figure 8 montre l'évolution de la commande en % qui doit être donné à la vanne en fonction de E la différence de température, et DE la dérivée.

Après développement de la commande floue, nous nous sommes rendu compte que le générateur de code PLC Coder ne fonctionne pas avec un bloc de commande floue. Nous avons donc gardé la commande par PID.

1.5 LES DIFFICULTES RENCONTREES

Les difficultés majeures rencontrées dans cette partie concernait l'établissement du modèle mathématique. La modélisation de la résistance thermique du radiateur n'est pas linéaire au débit contrairement à une résistance électrique.

- Résistance électrique : $U = R.I$
- Résistance thermique : $(T_{salle} - T_{Radiateur}) = f(Débit_{eau}, \Delta T_{eau}(Débit_{eau}))$

ΔT_{eau} est fonction du débit d'eau, et correspond à la différence entre la température de l'eau en entrée et celle en sortie. Nous avons choisi de travailler en Puissance thermique afin de faciliter la compréhension des échanges qui se passe dans notre modèle lors de la simulation. Et il se trouve que, comme indiqué plus haut, la puissance thermique s'écrit :

- $Q_{fournie} = Q_{Nominale}\left(\frac{\Delta T}{60}\right)^n$ $avec\ \Delta T = \frac{T_d + T_r}{2} - T_s$

Comme on peut s'en rendre compte, nous avons abouti à un modèle mathématique complexe, qui nous a créée bon nombre de problèmes par la suite (résultats peu physiquement probables, incohérence des valeurs obtenues…)

Une autre difficulté majeure était « l'effet retard » du radiateur, lorsque la vanne est fermé, la température du radiateur redescend instantanément à la valeur de consigne, mais par manque de temps, nous n'avons pas cherché à résoudre l'erreur puisque l'erreur n'influe pas sur la consommation du radiateur, la consommation du radiateur étant la valeur qui nous intéresse le plus.

2 PARTIE COMMERCIALE

2.1 CAHIER DES CHARGES

Cette partie du projet s'est déroulée d'Octobre à Décembre et consistait à répondre au cahier des charges suivant :

- Analyser plusieurs solutions permettant la régulation thermique ;
- Faire des appels de devis auprès de plusieurs entreprises ;
- Analyser les devis afin de retenir uniquement l'offre la plus intéressante ;
- Passer commande auprès de l'entreprise et assurer le suivi.

2.2 LES SOLUTIONS PERMETTANT UNE BONNE REGULATION

Il existe peu de modèles d'électrovannes sur le marché actuellement, nous avons donc choisi la vanne Thermokon qui utilise le protocole radio innovant EnOcean. A partir de là, nous avons envisagé plusieurs solutions qui nous permettrait de mettre en place le banc de test :

- Un automate programmable qui communiquerait au moyen de port série et passerelle EnOcean ;
- Une raspeberry avec un dongle USB ⇔ EnOcean.

La 2nd solution est moins coûteuse mais il aurait fallu développer un programme en C, sans environnement de développement, et l'ensemble aurait fait « bricolage », c'est la raison pour laquelle nous avons opté pour la 1ère solution. De plus, la 2nde solution est difficilement généralisable à toute l'école.

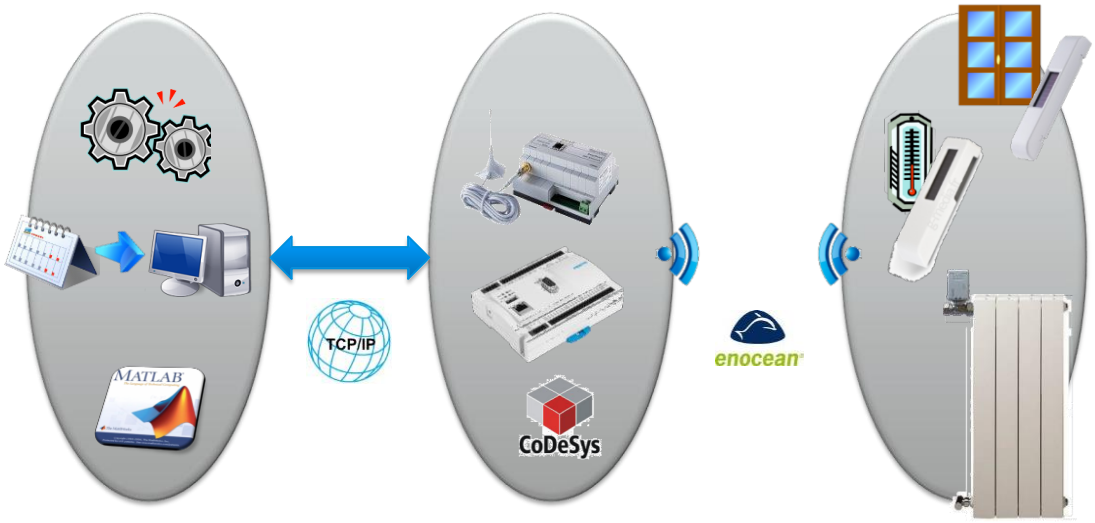


Figure 9: Synoptique du système retenu

La synoptique du système choisi décrit parfaitement bien le système sur lequel nous avons mis nos préférences.

2.3 APPEL D'OFFRE

Une fois le type de solution envisagée, nous avons contacté Festo et Wago. Festo proposait un service supplémentaire par rapport à Wago : l'élaboration d'un programme CoDeSys afin de ne programmer notre banc de test que sous Matlab. Ce service étant facturé 2 800 € (et développable par nous-mêmes), en ajoutant le surcoût des pièces par rapport à Wago de 1 200 €, nous avons finalement retenu l'offre de Wago pour 1 220 € environ (1.1Bon de commande pour Wago).

2.4 DIFFICULTES RENCONTREES

La ''seule'' difficulté que nous ayons rencontrée, fut la prise de décision car nous savions que Polytech entretenait un bon partenariat avec Festo. C'est une des premières fois que le pôle automatique choisit un concurrent à Festo. Mais la différence de prix de près de 4000 € fut largement décisive.

3 MISE EN PLACE DU BANC DE TEST

3.1 CAHIER DES CHARGES

Après avoir réalisé l'étude théorique, il a fallu mettre en pratique les résultats de régulations que nous avons obtenus en simulation, et les programmer sur le matériel reçu. Les principales tâches à réaliser concernant cette partie sont :

- Mettre en place et tester l'équipement
- Se familiariser avec l'environnement CoDeSys et la librairie EnOcean
- Mettre en place le programme de régulation
- Stocker les paramètres du système
- Réaliser une interface graphique pour visualiser les paramètres en temps réel
- Rédiger un manuel d'utilisation pour la suite du projet

3.2 OUTILS MATERIEL ET LOGICIEL UTILISES

3.2.1 Matériel Wago

Les équipements utilisés sont des équipements Wago, muni de la technologie EnOcean.

EnOcean est une technologie sans fil innovante, basée sur des émetteurs radio à très faible consommation.

Pour la réalisation de notre banc d'essai nous avons donc utilisé le matériel suivant :

Référence	Description	Utilisation
750-881	API Contrôleur Ethernet 10/100	Contenir l'ensemble des programmes
750-653/003-000	Borne d'interface série RS 485	Communication série entre la passerelle et l'automate
750-600	Borne d'extrémité finale de bus	Fermeture du circuit
787-1002	Alimentation à découplage 230 V AC / 24 V DC 1.3A	Alimenter l'automate et la passerelle en 24V – 1.3A
5102-9640	Récepteur-émetteur EnOcean Passerelle communication série RS 485	Conversion et transmission des données séries en données radio
5109-7084	Electro-Vanne à piles communication EnOcean	Régulation du débit dans le radiateur
5109-7096	Capteur de température sans pile EnOcean	Mesurer la température ambiante
5109-7786	Capteur de fenêtre EnOcean	Détecter l'état d'ouverture de fenêtre

3.2.2 Environnement de programmation : CoDeSys

La programmation sur le contrôleur programmable WAGO-I/O-SYSTEM utilise l'environnement CoDeSys. CODESYS est un environnement de développement pour des automates programmables industriels (API) selon le standard CEI (Commission électrotechnique internationale) pour le

développement d'applications dans l'automatique industrielle, il est aussi libre de droits et peut être installé et utilisé sur tout ordinateur. Les langages inclus dans CoDeSys sont tous les langages spécifiés de la CEI.

3.3 INSTALLATION MATERIELLE ET LOGICIELLE

La documentation fournie par Wago est assez claire et détaillée pour nous avoir permis de réaliser l'installation et la configuration de l'équipement. Le schéma global de notre banc de test est le suivant :

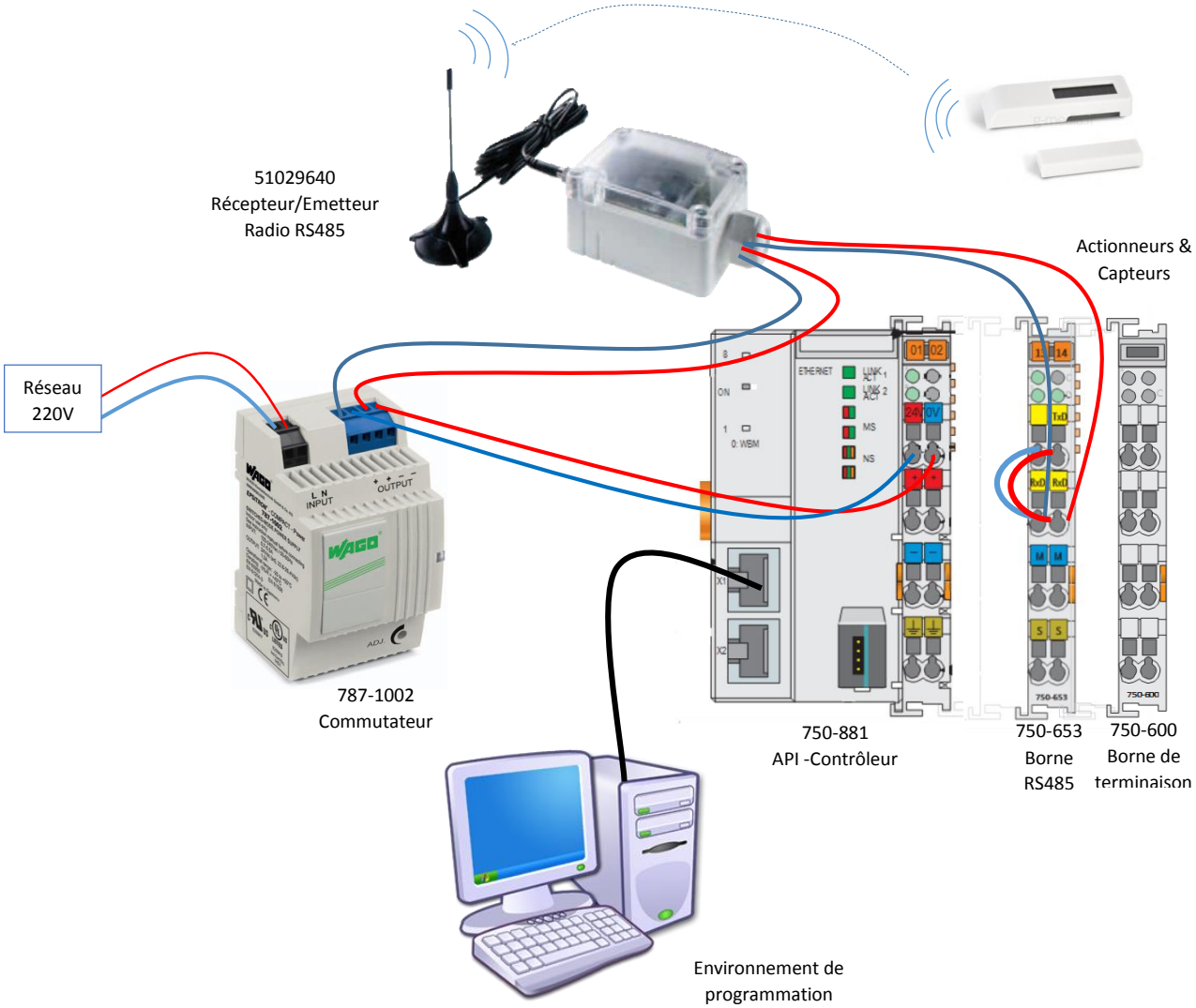


Figure 10Schéma global du système monté

L'API est connecté d'une part à l'ordinateur via une connexion Ethernet, et de l'autre part à la passerelle EnOcean via sa borne de liaison série RS485. La borne de liaison est montée sur rail avec le contrôleur, suivie de la borne de terminaison qui sert à fermer le bus interne et garantir une circulation correcte des données. Quant à la passerelle, elle communique en émission et réception Radio EnOcean avec les capteurs et l'actionneur de notre système. Enfin, l'alimentation se fait via le commutateur qui transforme la tension alternative du réseau en tension 24V continue nécessaire au circuit.

WAGO dispose d'une bibliothèque de fonctions complète pour la gestion des capteurs sans fil. Afin de simplifier le travail d'intégration, les blocs fonctionnels décodent les données des trames radio et placent les valeurs reçues dans des variables de sorties adaptées au type de profil du capteur.

Nous avons dans un premier temps mis en place un programme de test pour vérifier la bonne mise place des équipements, se familiariser avec l'environnement de programmation et les différentes fonctions de la bibliothèque EnOcean.

Le programme réalisé récupère les identifiants (Adresse MAC) de chacun des périphériques, et communique avec eux, via la passerelle, en utilisant une variable de type EnOcean contenant toutes les données des trames reçues ou à transmettre. Ensuite, le programme lit les relevés des capteurs, et envoie un ordre à la vanne.

Nous avons vérifié suite à ce programme de base certains paramètres qui nous ont permis par la suite de contrôler et superviser notre système de régulation qui sont :

- les temps d'envoie des trames de données provenant des capteurs : 15 mn,
- le temps de réaction de la vanne à un ordre : 10mn,
- La vanne fournie des données concernant la capacité de la batterie ou l'état de son clapet, et signal toute défection interne à l'actionneur.

3.4 MISE EN PLACE DU PROGRAMME EMBARQUE

3.4.1 Modèle du programme

Le programme créé consiste à réguler la température d'une salle en agissant sur la vanne de régulation selon le modèle globale suivant :

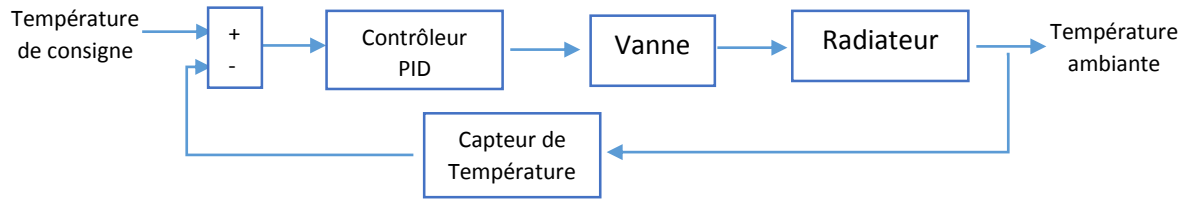


Figure 11: Schéma fonctionnel du système

L'algorithme commence par récupérer toutes les données du système. Ensuite, vérifie si la valeur de la température ambiante correspond à la valeur désirée. Enfin, donne l'ordre de position de vanne à travers un régulateur PID.

Il prend aussi en compte certains paramètres tels que l'état de la fenêtre : si celle-ci est ouverte, la vanne est automatiquement fermée pour éviter les pertes d'énergies. Et enfin, le programme stocke les données des variables afin de surveiller l'état du système.

3.4.2 Régulation

Pour programmer le bloc de régulation, nous devions générer du code CoDeSys à partir de Matlab Simulink afin de récupérer les paramètres du contrôleur PID que nous avons étudié. Ce qui nous a amenés à utiliser l'outil PLC Coder qui génère du texte structuré à partir de modèles Simulink et de code MATLAB embarqué, puis utilise un environnement de développement intégré (IDE) pour compiler le code et l'exécuter sur un périphérique PLC.

Simulink PLC Coder est un outil qui génère du texte structuré CEI 61131 pour des automates programmables industriels (PLC et PAC). Il permet d'utiliser une approche Model-Based Design pour des équipements contrôlés par des automates programmables.

3.4.3 Interface de Visualisation CoDeSys

Afin de pouvoir visualiser, consulter et utiliser les données de l'automate programmable avec CoDeSys, le système de programmation intègre un éditeur de visualisation.

Nous avons réalisé un objet de visualisation sur CoDeSys qui permet d'accéder aux variables et avoir des informations sur leur état en mode **En ligne**.

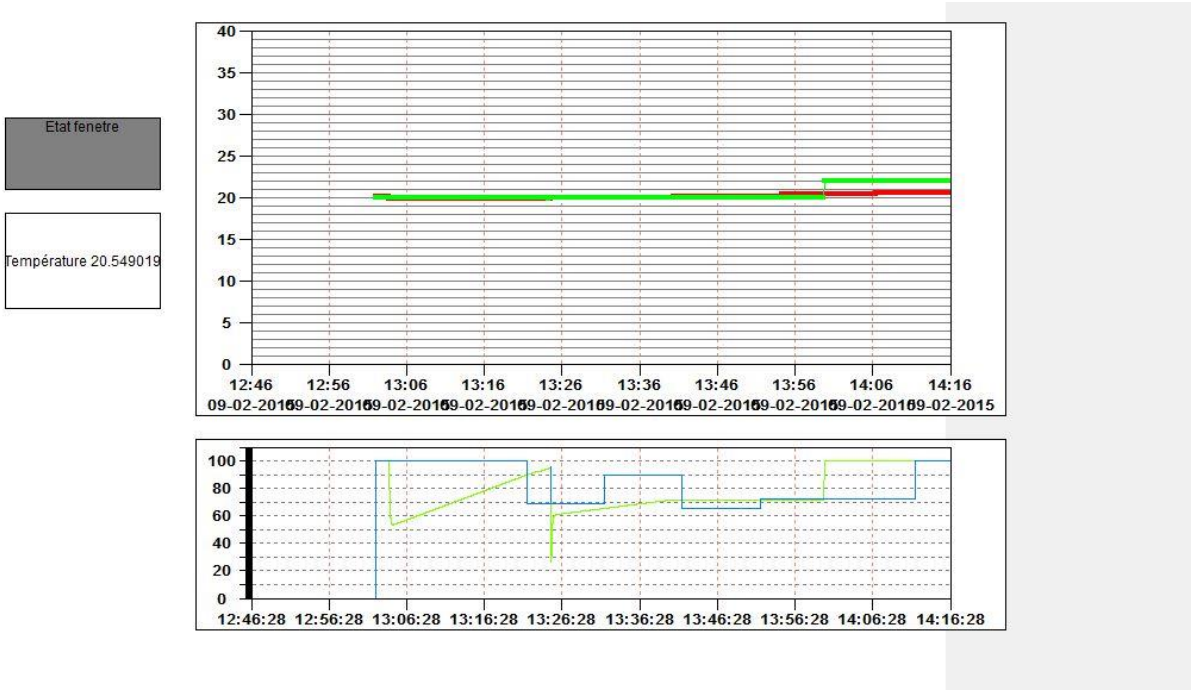


Figure 12: Visualisation sur CoDeSys

L'image ci-dessus représente notre page de visualisation sous CoDeSys, sur laquelle on peut voir l'état de la fenêtre sur un encadré qui change de couleur à chaque changement d'état, la valeur de la température relevée par le capteur de température, ainsi que l'évolution de la valeur de température (en rouge) comparée à la température de consigne (en vert) sur le graphique de haut, et l'évolution de la position de la vanne (en bleu) comparée à l'ordre de positionnement de la vanne (en vert). On remarque aussi le déphasage en la consigne et la position de la vanne de 10 minutes. La vanne émet et reçoit toutes les 10 minutes.

Ceci nous a permis de consulter les variables du programme, mais uniquement en mode « en ligne » car les valeurs sont perdues dès la mise hors connexion. Pour afficher une courbe sur une journée, on doit garder un poste allumé en permanence sur la journée entière. Ainsi nous avons décidé de stocker les données afin de pouvoir les visualiser sur un autre support en utilisant le protocole TCP/IP Modbus ou une HTTPRequest (nous avons opté pour le protocole modbus).

3.4.4 Module de stockage

Afin de pouvoir récupérer les valeurs des variables avec notre programme en utilisant le protocole TCP/IP Modbus, nous avons eu besoin de spécifier l'adresse de stockage de chacune des variables qu'on souhaite stocker, en prenant en compte le type de chacun et surtout sa taille, afin de

pouvoir spécifier s'il s'agit d'un Bit pour les booléens par exemple, un Byte (1 octet), un Word (2 octets) pour les entiers, ou un Double Word (4 octets) pour les réels.

Une fois les adresses attribuées, notre module de stockage enregistre, à chaque changement de position de vanne, sinon chaque demi-heure, sur une durée de 3 jours, les valeurs des variables suivantes:

- la température de consigne
- la température relevée,
- l'état du capteur de fenêtre,
- la position de consigne de la vanne
- la position actuelle de la vanne
- la date et l'heure du relevé
- l'état des variables d'erreurs

3.5 REALISATION DE L'INTERFACE GRAPHIQUE

L'interface sert principalement à récupérer les données stockées sur l'automate, afin de pouvoir surveiller la bonne mise en route du programme et superviser notre système, à n'importe quel moment.

Pour réaliser notre interface graphique, nous avons utilisé le Framework .Net. Ce Framework est une plateforme de développement largement utilisée pour la création d'applications. Elle comprend les langages de programmation C# et Visual Basic, le Common Language Runtime, ainsi qu'une abondante bibliothèque de classes.

Dans un premier temps, nous avons développé un petit programme C# afin de communiquer avec l'automate avec le protocole TCP/IP Modbus et pouvoir lire et écrire des variables sur la mémoire de l'automate. Ensuite, nous avons opté pour la librairie WPF (Windows Presentation Foundation) pour sa capacité à développer une application aussi bien à l'aide du balisage que du code-behind, afin de réaliser une interface de visualisation de courbes d'évolution de la température et de la position de la vanne, ainsi qu'un rapport d'erreur qui permettra de superviser le système.

3.5.1 Développement du programme NModbus

Nous avons donné à ce programme le nom de la bibliothèque utilisée pour communiquer avec l'automate au moyen du protocole TCP/IP Modbus. Ce protocole permet de récupérer les valeurs de données en passant les adresses en paramètre :

```
using (TcpClient client = new TcpClient(ipAddress, 502))
{
      ModbusIpMaster master = ModbusIpMaster.CreateIp(client);
      //assign Modbus-Startaddress and how many register should be read
      int test = Convert.ToUInt16(address) + 12288;
      ushort StartAddress = Convert.ToUInt16(test);

      // Read Input register values
      ushort[] values = master.ReadHoldingRegisters(StartAddress,1);
      this.data = (int)values[0];
}
```

L'extrait de code ci-dessus est un échantillon de code utilisé dans notre programme NModbus pour récupérer un mot (=2 octets). Il nous a fallu par la suite modifier cet échantillon pour pourvoir

récupérer plusieurs octets ou des double mots (4 octets = 32 bits) et convertir au bon format (réel, float, int…).

Figure 13: Aperçu de Nmodbus

Cette interface graphique Nmodbus nous permet de lire des mots (2 octets) et des réels (4 octets), spécifier le nombre de valeurs à lire, et écrire des mots ou des réels à des adresses spécifiées. Ce programme a été développé en quelques jours et était vraiment conçu pour pouvoir vérifier de façon simple et rapide que la régulation fonctionnait comme nous le souhaitions.

3.5.2 Développement de ThermoVisu :

Cette interface utilise la même librairie et des fonctions similaires au programme NModBus pour récupérer tous les tableaux de variables stockés sur l'automate, et les afficher sous forme de courbe.

Pour l'affichage de courbes, nous avons utilisé le WPF Toolkit, qui est une boite à outil présentant une suite de fonctionnalités et composants WPF mis à disposition, elle permet de mettre en place des courbes et des graphiques sur une solution WPF.

Nous avons aussi intégré un rapport d'erreur qui affiche tous les problèmes qui peuvent survenir durant l'exécution du programme, que ce soit une perte de connexion radio avec l'un des capteurs, ou une mauvaise manipulation de la vanne (obstruée, bloquée…). Les données sont ensuite exportées si besoin dans un fichier texte, après avoir défini le chemin d'exportation.

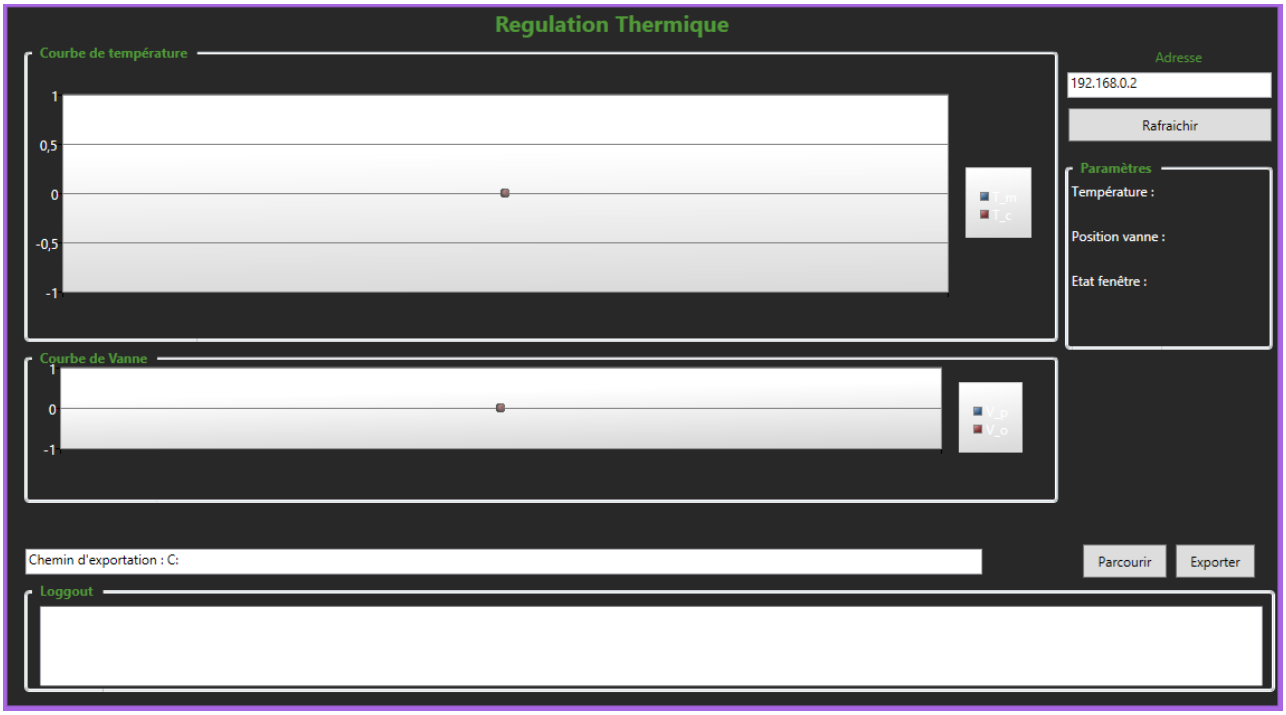


Figure 14: Interface graphique ThermoVisu

3.6 PROBLEMES RENCONTRES

Durant la mise en place du banc de test, nous avons rencontré des difficultés parmi lesquels :

- absence visibilité sur l'état d'alimentation des capteurs : Ces capteurs sont des capteurs sans pile alimenté à l'aide d'une cellule solaire et d'un accumulateur d'énergie interne, mais qui ne fournissent aucune information sur leur alimentation. Pendant la phase de tests des équipements, nous n'arrivions pas à récupérer les valeurs mesurées par les différents capteurs car ils n'étaient pas alimentés, sans savoir aussi que le temps d'envoi d'une trame est de 15mn (contrairement aux 10 minutes annoncés dans la datasheet), ce qui nous a causé un retard sur notre avancement pour la recherche d'un problème inexistant.
- La mise en place du bloc PID fourni par une bibliothèque CoDeSys ne remplissait pas complètement sa fonction car la consigne n'était jamais atteinte, nous avons donc été amenés à utiliser le bloc PID réalisé en simulation sur Matlab. PLC coder nous a permis de générer un code en langage structuré à partir du bloc fonctionnel sous Simulink.
- La visualisation sous CoDeSys requérait de rester en mode « En ligne » sur toute une journée afin de pourvoir visualiser une courbe sur 24h. La solution pour laquelle nous avons opté fut de stocker les données et de les afficher sur une application graphique.
- La réalisation de l'interface graphique sur Visual studio demande des connaissances en langages C#, un langage que nous n'avions pas eu l'occasion d'utiliser avant, ce qui a représenté une difficulté pour nous et a ralenti notre avancement.

3.7 RESULTATS OBTENUS

3.7.1 Rendu

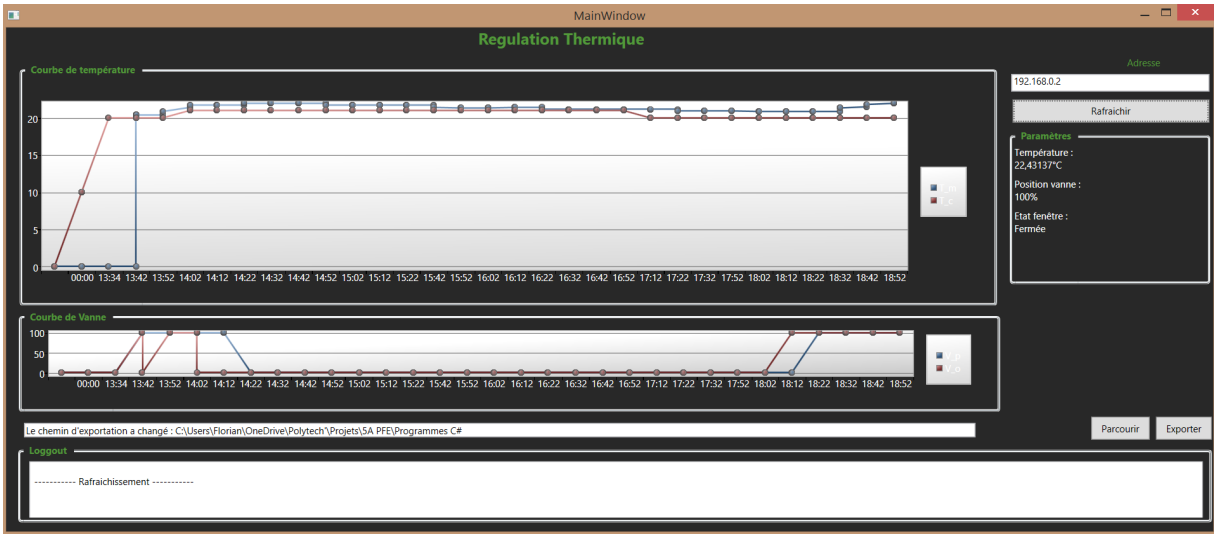
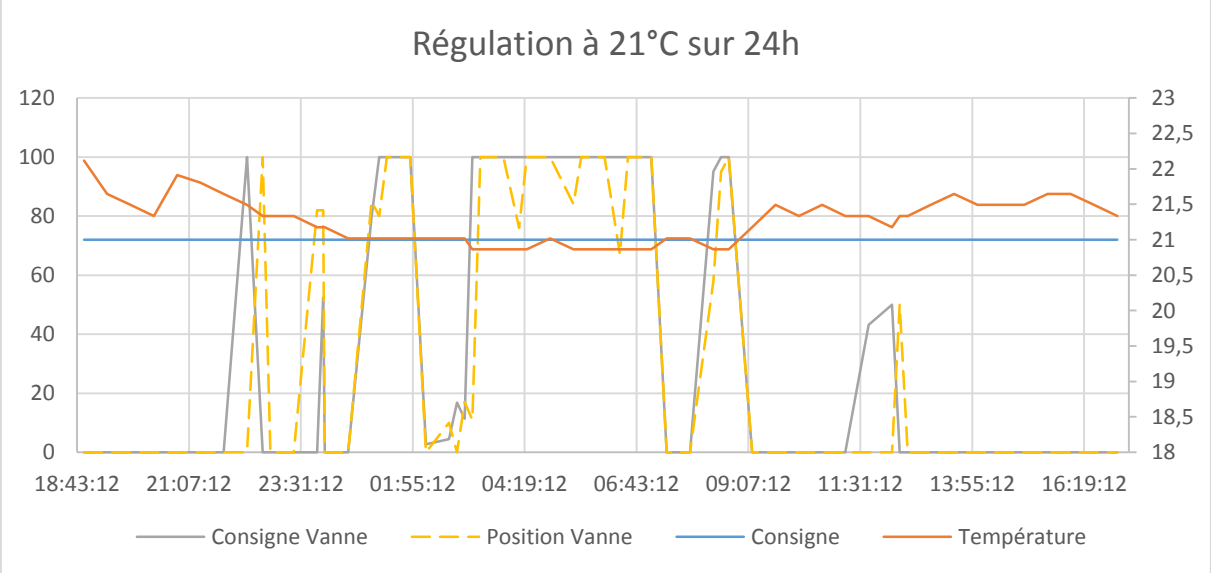


Figure 15: Interface ThermoVisu (plus de détails en Interface ThermoVisu)

3.7.2 Aboutissement du banc de test

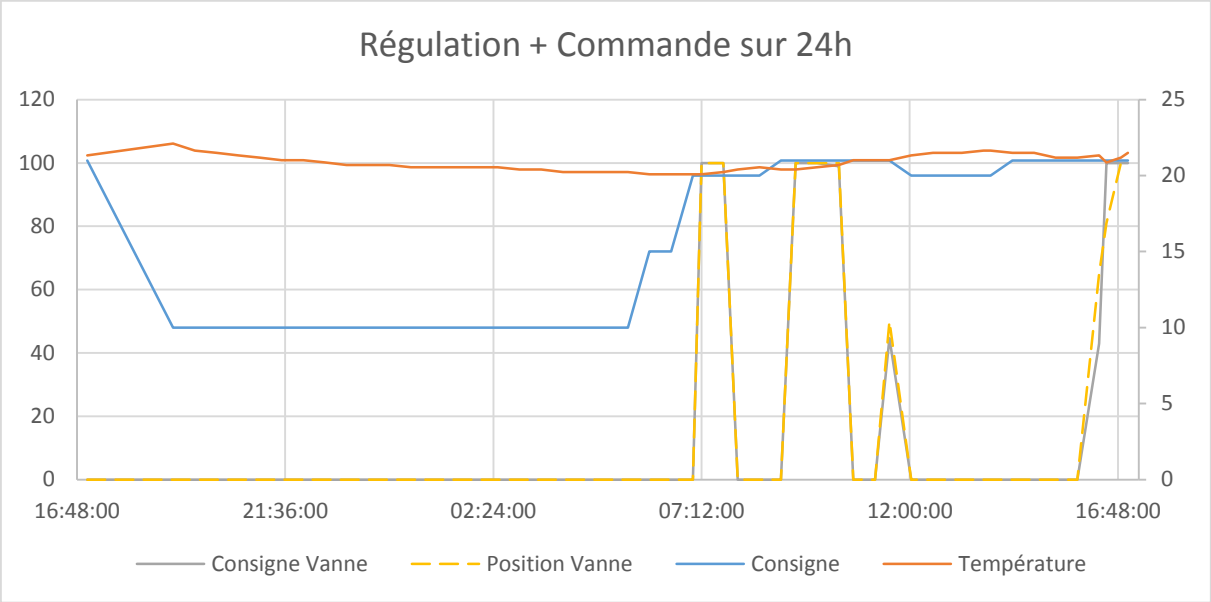
In fine, nous avons pu passer à l'étape de tests finaux et de confrontation de nos résultats. Un premier test sur une journée nous a permis de visualiser la régulation puisque la consigne était de 21°C sur les 24h. Les graphes sont obtenus grâce à notre programme ThermoVisu, on exporte les données en fichier .txt que l'on traite ensuite avec une macro sur Excel.



On s'aperçoit que :

- L'erreur est assez faible entre 0.5°C et 1°C ;
- Le système est robuste puisqu'il permet de stabiliser la température proche la consigne ;
- La moyenne d'ouverture de la vanne sur une journée est de 35,92 %.

Un autre test sur 24h avec commande nous permit de visualiser les variations lorsque la consigne change :



Ce 2nd test nous permit principalement de relever le pourcentage moyen d'ouverture de la vanne sur une journée, qui est de 16,24 %.

3.7.3 Rendement

En comparant avec le premier test, on s'aperçoit que les économies d'énergie sont surtout réalisées pendant la nuit. Au final le rendement ''estimé'' est de :

$$\eta = \frac{moyenne(ouverture\ de\ la\ vanne\ système\ commandé)}{moyenne(ouverture\ de\ la\ vanne\ système\ non\ commandé)} * 100 = 45{,}222\%$$

On aboutit à un résultat mieux que la théorie puisque nos calculs sous Matlab nous permettaient d'estimer le rendement à 34%.

Nous insistons sur le caractère estimé du rendement, puisque la température de la salle ne dépend pas que des radiateurs. Elle dépend notamment du nombre de personnes qui l'occupent, de la température extérieure, porte ouverte ou non…

CONCLUSION

La réalisation de ce projet de fin d'étude nous a permis de mettre en pratique la majorité des connaissances acquises pendant nos trois années de formation en IMA, notamment les cours d'automatique et d'informatique. Nous avons également acquis des notions en gestion de projet et négociation commerciale, à travers les différentes parties du projet.

Le but du projet étant de construire un démonstrateur de régulation thermique permettant de minimiser les pertes d'énergie, nous arrivons à réaliser une économie de 45%, un pourcentage que nous estimons capable de remplir notre objectif.

Le banc de test mis en place permet d'établir de bonnes bases pour le déploiement de cette solution à Polytech Lille. On pourrait par la suite envisager :

- une amélioration de la régulation (régulation par apprentissage) ;
- une optimisation de la valeur de température de consigne en prenant en compte la réservation de la salle ;
- une externalisation des variables de sortie de l'automate dans une base de données (les différentes erreurs, historique de température et position de vanne) ;
- une supervision de la température dans l'ensemble des salles…

Enfin, nous avons écrit une documentation (téléchargeable ici) détaillant le matériel Wago, le logiciel CoDeSys, ainsi qu'une explication de notre programme de régulation, dans le but d'assurer un suivi du projet.

1 ANNEXES

1.1 BON DE COMMANDE POUR WAGO

POLYTECH LILLE

DEMANDE D'ACHAT

Ecole Polytechnique Universitaire de Lille
Bd Paul Langevin
59655 VILLENEUVE D'ASCQ CEDEX

DATE : 23/02/2015

Les factures, établies en 3 exemplaires, sont à adresser directement à
Université des Sciences et technologies de Lille
Service facturier - Bâtiment A3 Cité Scientifique
59655 VILLENEUVE D'ASCQ Cedex
Le prix doit être arrêté, toutes taxes comprises,
Une facture ne peut faire référence à plusieurs bons de commande, et doit toujours comporter le numéro de commande

les marchés à votre disposition — A remplir par le Demandeur

BON DE COMMANDE A ENVOYER PAR MAIL à :
commande-fr@wago.com

N° de devis associé : LFU01160-3

FOURNISSEUR

Nom : WAGO
Adresse : Paris Nord 2
83 rue des Chardonnerets
93290 Tremblay-en-France
Tél : 01 48 17 54 03
Fax :
à l'attention de : Magali BODIN

DEMANDEUR (pour livraison)

POLYTECH LILLE
AVENUE PAUL LANGEVIN
59650 VILLENEUVE D'ASCQ
Service : Pole Automatique
Demandeur : A.L. Gehin

Téléphone pour livraison: 03.28.76.73.00
Horaires de livraison : 8H30-12H00 / 13H30-17H00

Références	Fournitures	CODE NACRES	Quantité	Prix unitaire	Montant H.T.	Taux TVA	T.V.A	Montant T.T.C.
787-1002	Alimentation à découpage COMPACT 230 V AC / 24 V DC - 1.3 A	RB.01	1	40,73	40,73	20,00%	8,15	48,88
750-881	Contrôleur Ethernet TCP/IP 10/100MB + switch 2 ports	RB.01	1	347,71	347,71	20,00%	69,54	417,25
750-653/003-000	Borne interface RS 485 Configurable	RB.01	1	182,55	182,55	20,00%	36,51	219,06
5102-9640	RECEPTEUR/EMETTEUR RADIO RS485 IP42	RB.01	1	206,62	206,62	20,00%	41,32	247,95
750-600	Borne d'extrémité finale Bus	RB.01	1	10,12	10,12	20,00%	2,02	12,15
5109-7084	REGUL RADIO P.RADIATEUR M30x1,5 STD	RB.01	1	111,93	111,93	20,00%	22,39	134,32
5109-7088	SYSTEME ANTIDEMONTAGE P.VANNE RADIO	RB.02	1	10,04	10,04	20,00%	2,01	12,05
5109-7096	CAPTEUR DE TEMPERATURE COMPACT	RB.02	1	58,41	58,41	20,00%	11,68	70,09
5109-7786	CAPTEUR FENETRE/PORTE SS BATT.COMPAC	RB.02	1	52,36	52,36	20,00%	10,47	62,83
						Remise :		
						Montant H.T.	1 020,47	
						Montant TVA :	204,09	
						Montant TTC :	1 224,57	

DESTINATION — A remplir par le Responsable du centre de coûts

Formation

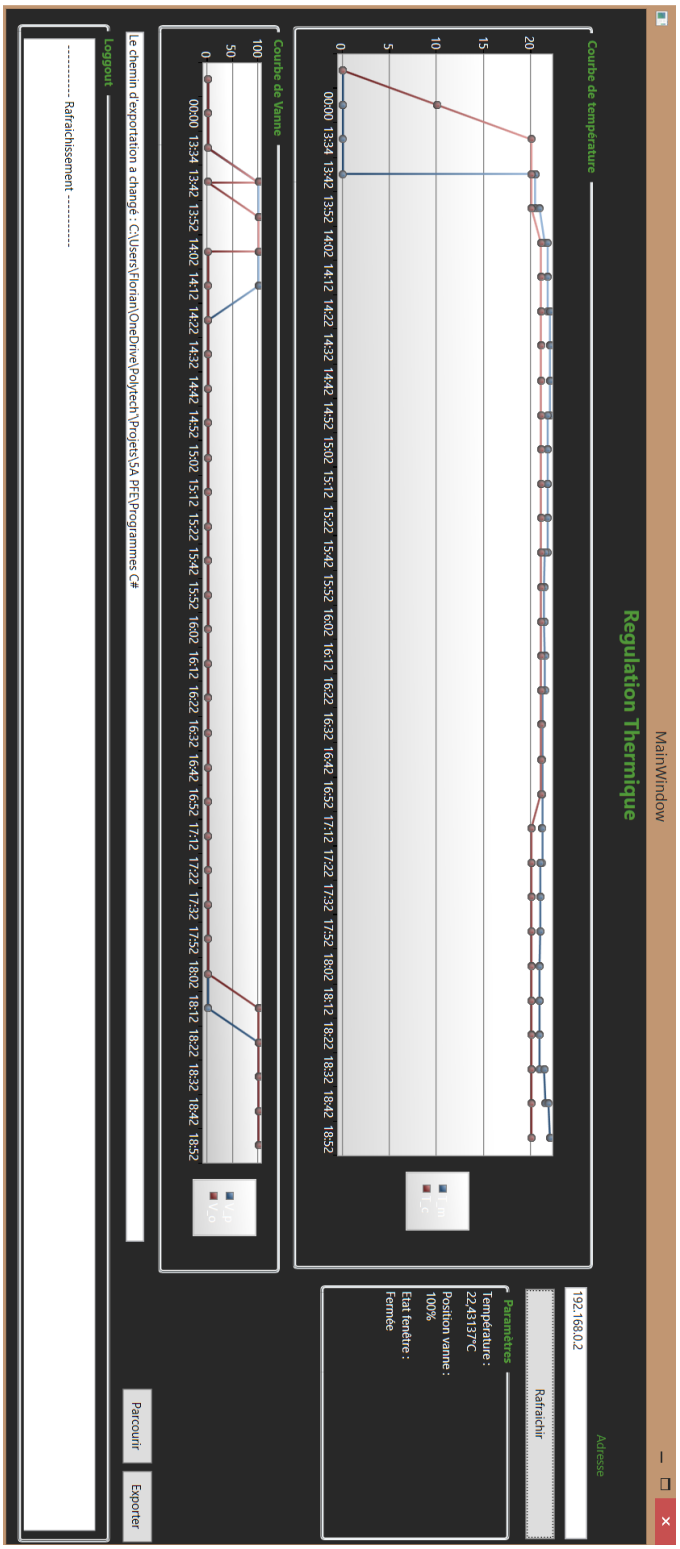
Ingénieur ☑ CF92142
PEIP ☐ CF92142
Licence Master ☐ CF92143
Apprentissage ☐ CF92144

DEPARTEMENT ☐
POLE Automatique ☑

Services généraux/ administration ☐ CF9213
Documentation ☐ CF92161
Relations internationales ☐ CF92162
Relations Entreprises ☐ CF92163
Scolarité Etudes ☐ CF92164
MIT ☐ CF92165
Recherche ☐ CF9213
Communication ☐ CF92166

FONCTIONNEMENT ☐
EQUIPEMENT ☑
Si équipement préciser : Ligne :
Libellé :

1.2 INTERFACE THERMOVISU



1.3 EXEMPLE DE FICHIER .TXT OBTENU APRES EXPORTATION