Bachelier en informatique industrielle
Bloc 2

Domaine Sciences et Technologies
Charleroi

Automatisation 2

Rapport : boitier de régulation

2022 – 2023

Dereli Ali – Balhor Otman – Acharif Ayoub – Aoufia Samir

Table des matières

1. Introduction :

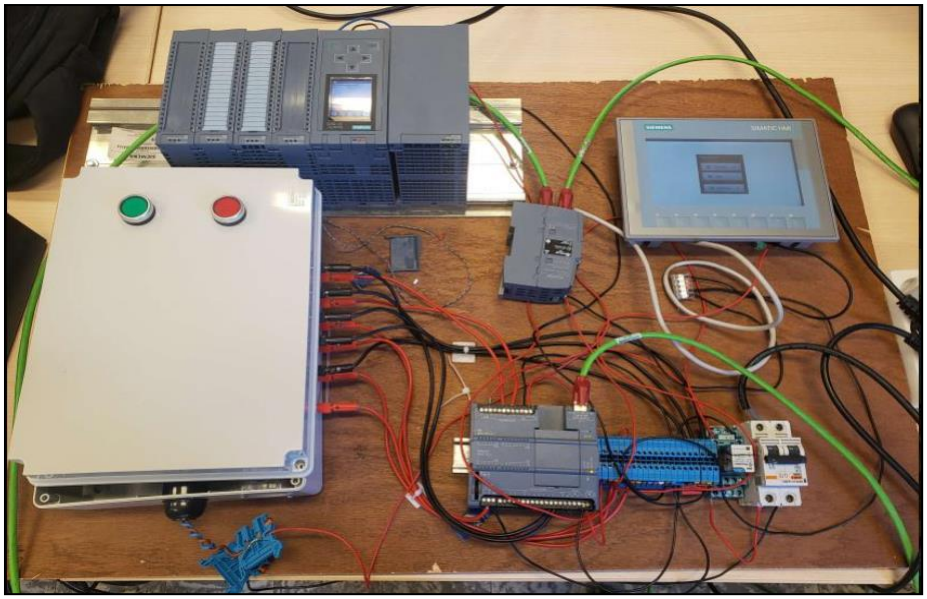
Pour le cours d'automatisation 2, il nous a été demandé de concevoir un programme qui consiste à réguler la température d'un boitier. Notre projet est composé de trois grandes parties. La partie réseaux, la partie supervision et la partie programmation.

2. Matériels :

La platine est composée de :

- Un automate 12/14 DC/DC/DC
- Un automate 15/16 F-3 PN
- Un boitier de régulation
- Un HMI KTP700 Basic PN
- Un relais
- Des borniers de raccordement

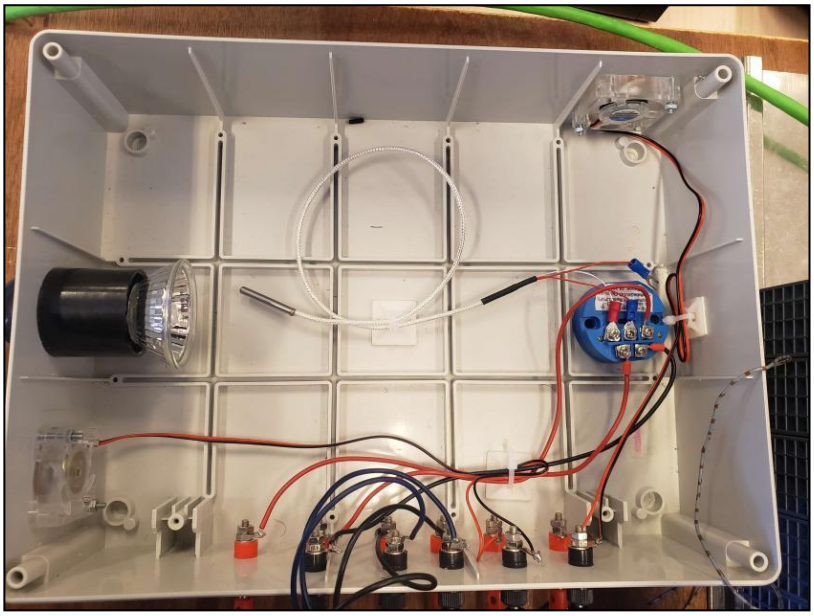
- Un switch
- Un disjoncteur



Composants de la platine

Le boitier de régulation est composé d':

- Un convertisseur 0-10V
- Une sonde PT100
- Deux ventilateurs
- Une lampe
- 2 boutons poussoirs (start, stop)



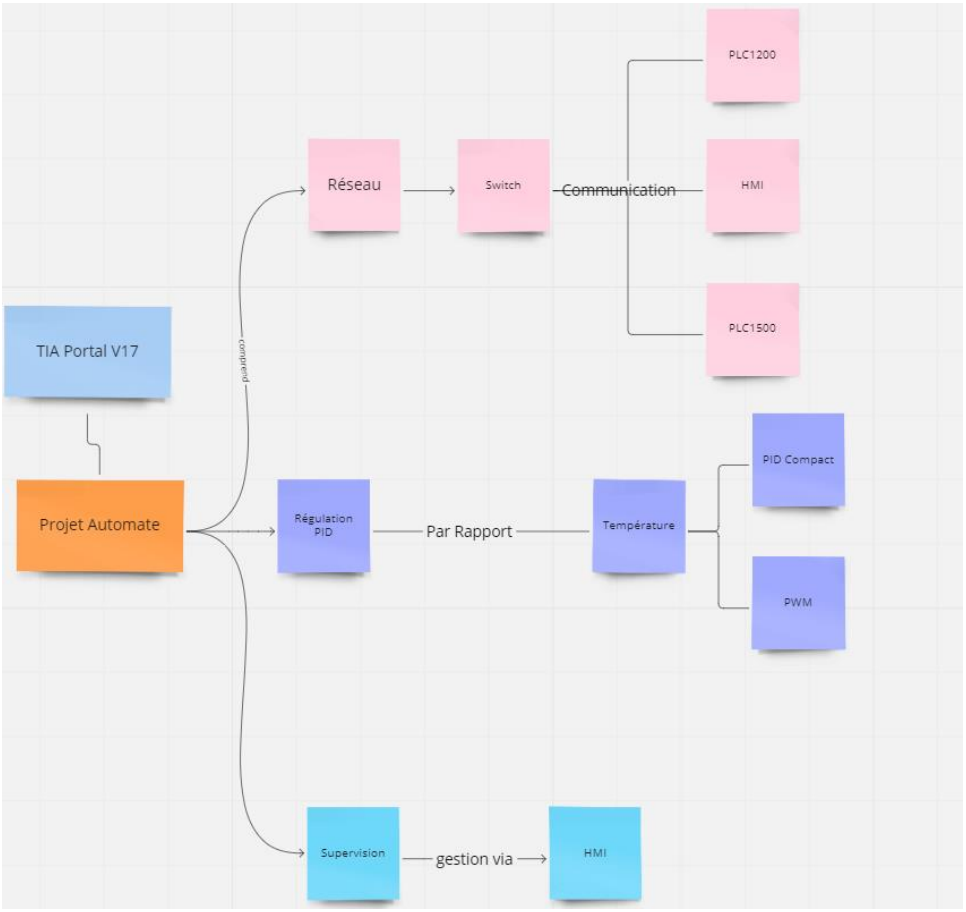
Composants du boitier de régulation

3. But du projet :

Notre objectif est de pouvoir réguler la température du boitier à l'aide des ventilateurs.la vitesse des ventilateurs devront également être régulé.

4. Mind Map :

Pour avoir une idée claire et avoir une vue plus compréhensible du projet global, nous avions commencé par établir un mind map. C'était toujours mieux de résumer les choses via des schémas et des cartes. Voici le mind map ci-dessous :



Mind Map du Projet Global

5. Schéma électrique :

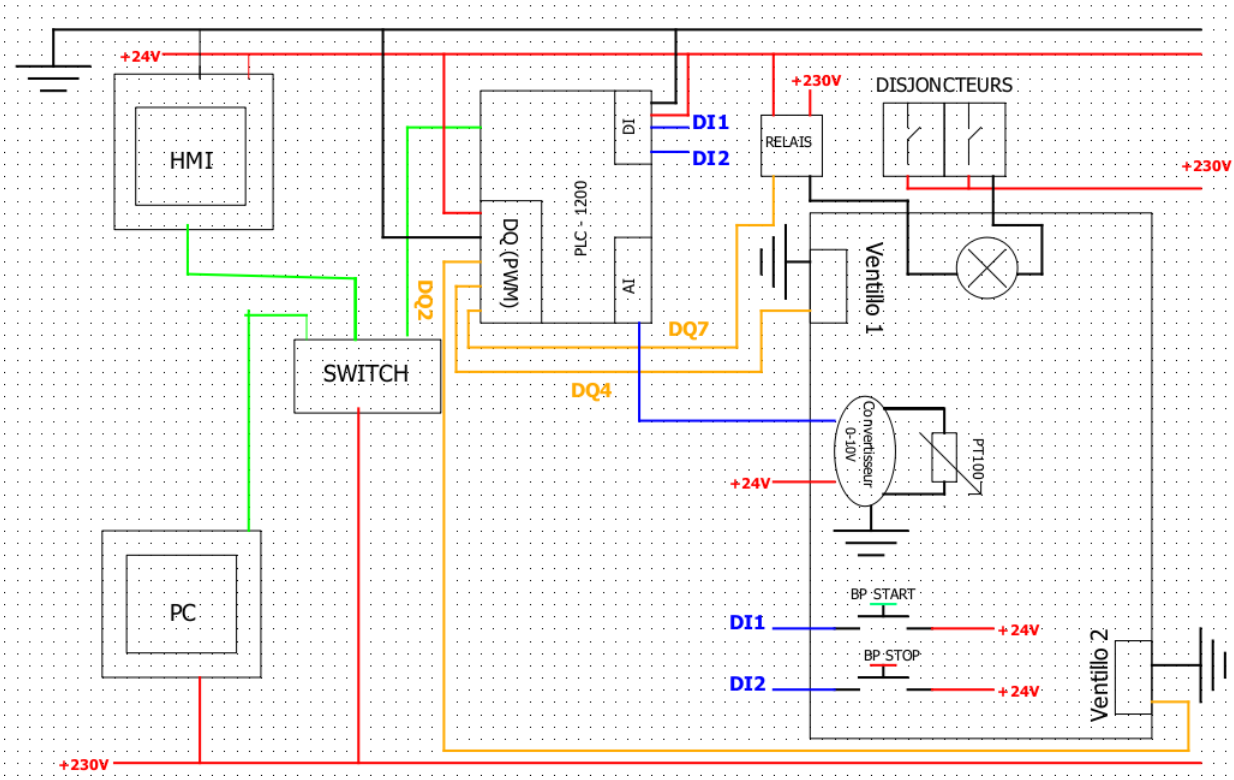
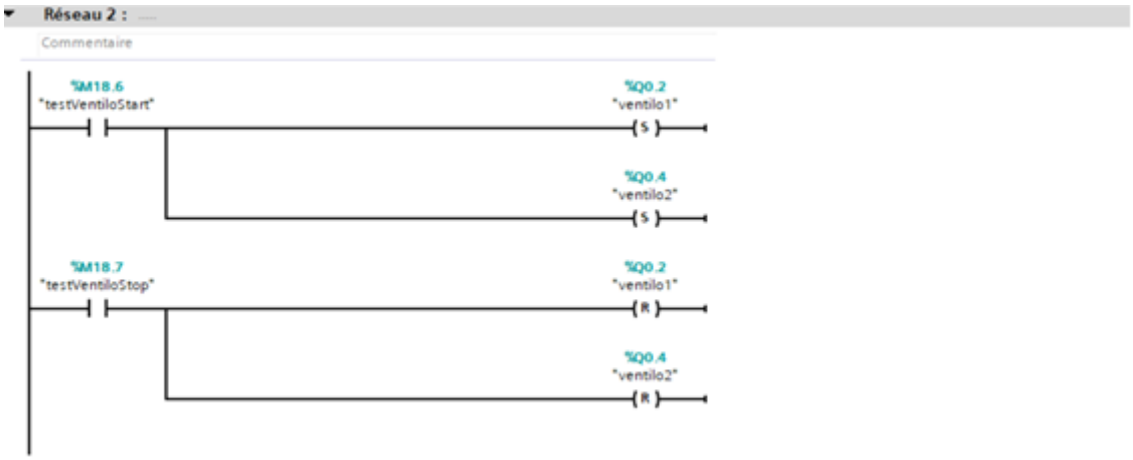


Schéma électrique

Pour dessiner ce schéma électrique, nous avons dû utiliser un logiciel nommé « QElectroTech » qui est gratuit.

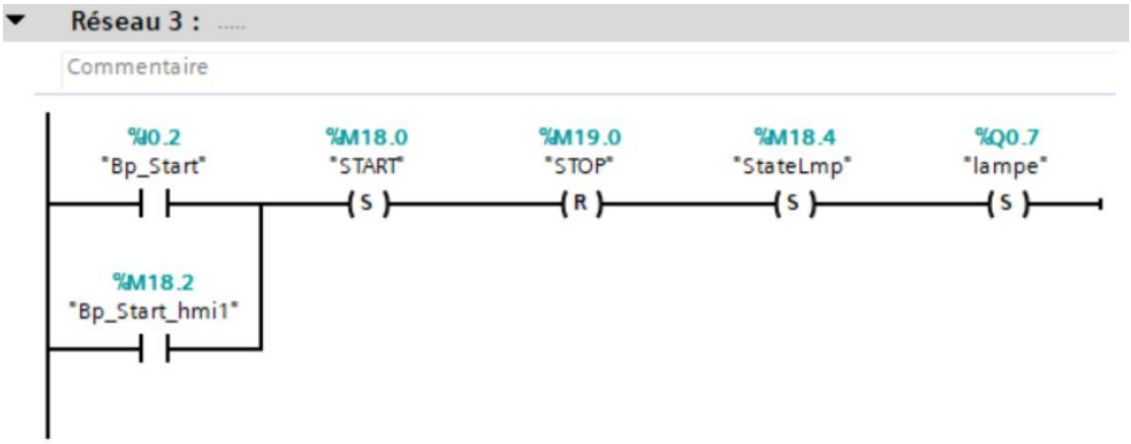
6. Le programme principal :

Pour réaliser la partie programme, nous avons dû utiliser des blocs fonctions ainsi que le langage ladder.



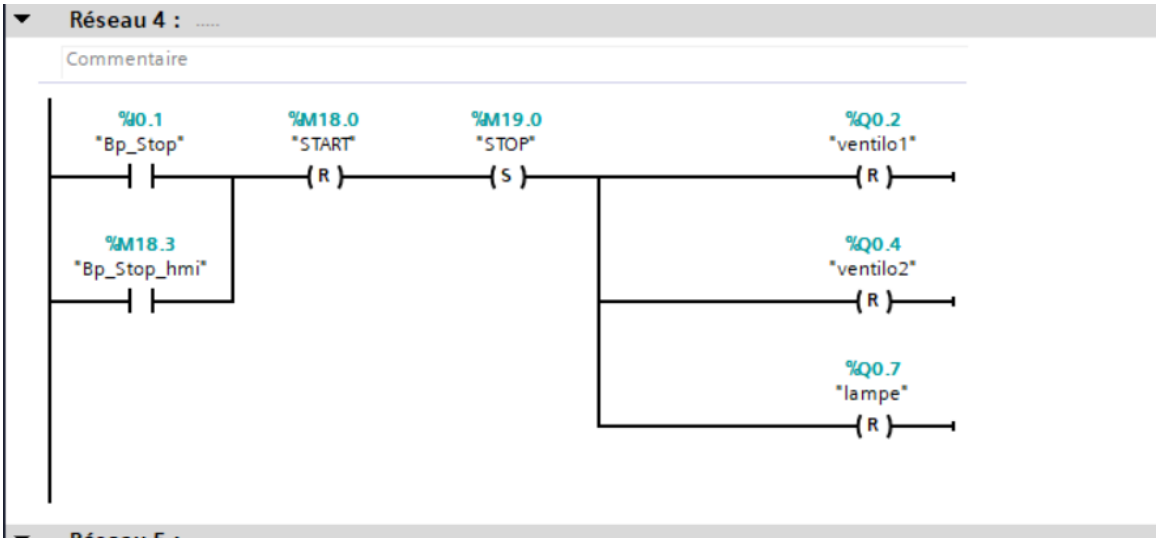
Test ventilateurs

Cette partie du ladder située au-dessus, nous permet de tester la vitesse des ventilateurs ainsi que leurs fonctionnements comme il nous a été demandé dans le cahier des charges. La platine doit être à l'arrêt pour pouvoir effectuer ce test.



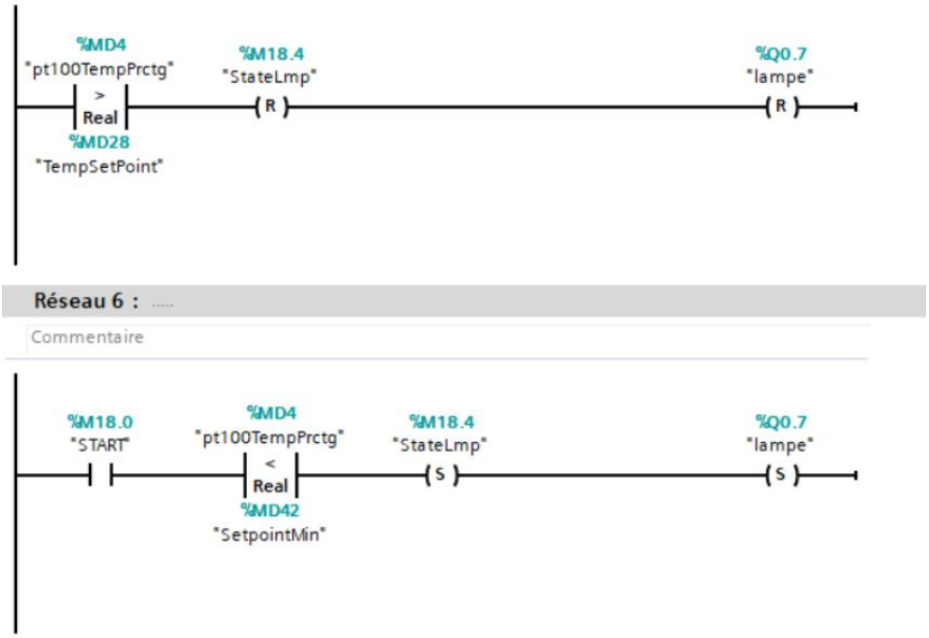
Ladder BP Start

Ce ladder nous permet de pouvoir allumer la lampe du boitier à l'aide d'un bouton START (bouton vert situé sur le boitier) ou à l'aide du bouton START situé dans le HMI. Lorsque nous appuyons soit sur le bouton physique ou celui de le HMI, le bouton STOP (bouton rouge situé sur le boitier) est mis à 0 (RESET) pour éviter qu'un conflit ait lieu entre ces boutons.



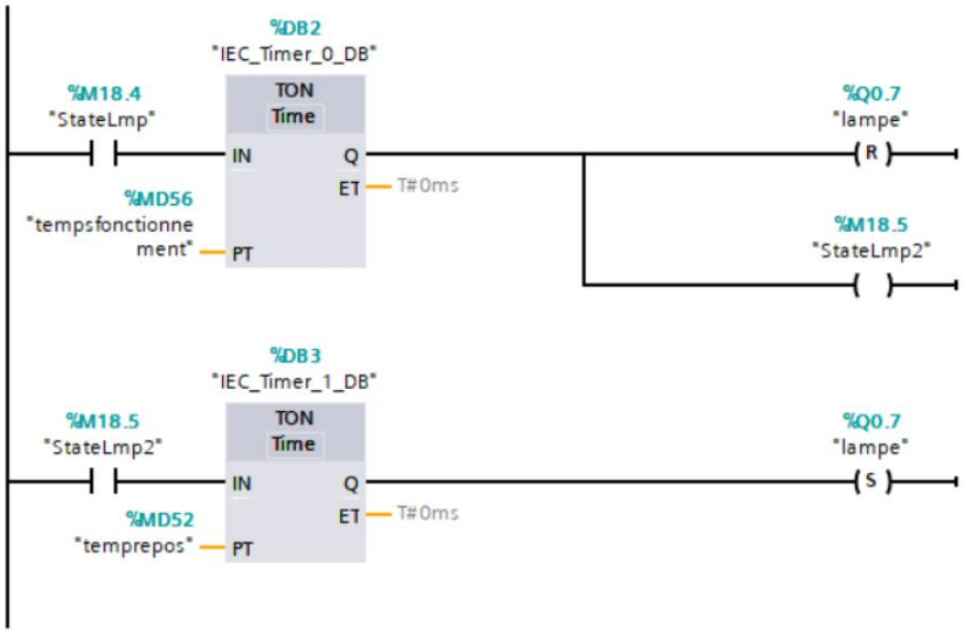
Ladder BP Stop

Ce programme ladder réalise la même fonction que le ladder du dessus mais pour le bouton STOP. Lorsque le bouton STOP, que ça soit le bouton physique ou celui situé dans le HMI, est activé, les deux ventilateurs et la lampe sont éteints automatiquement.



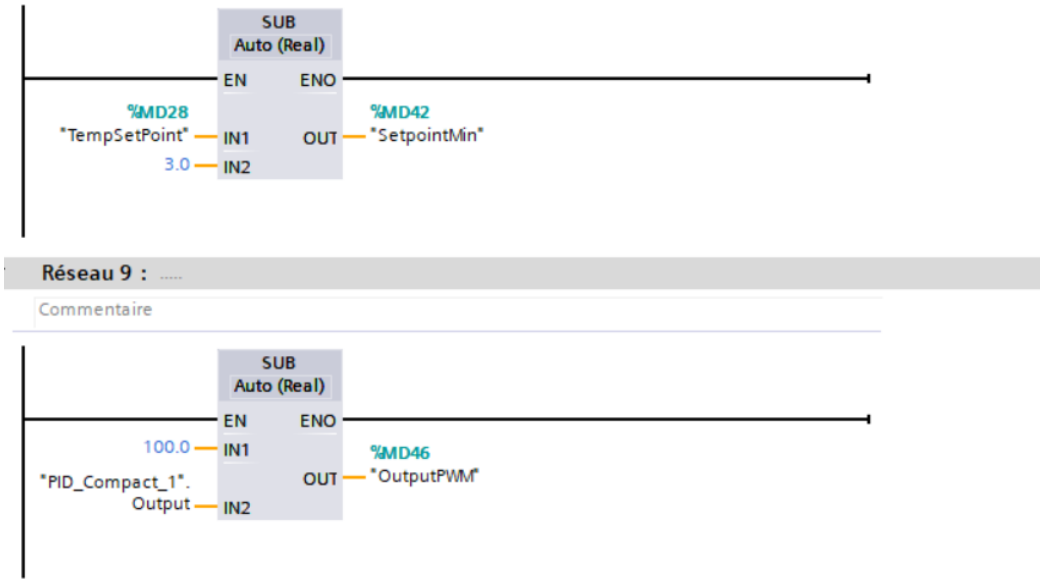
Ladder comparaison température

On utilise un bloc qui compare la température de la lampe (la valeur analogique) captée par la PT100 avec le « Setpoint » qui vaut 30°. Si la température de la lampe est supérieure au « Setpoint », alors la lampe sera éteinte. Cette dernière sera à nouveau active uniquement si la température est inférieure au « Setpoint ». L'état de la lampe dans notre ladder reviendra à 1.



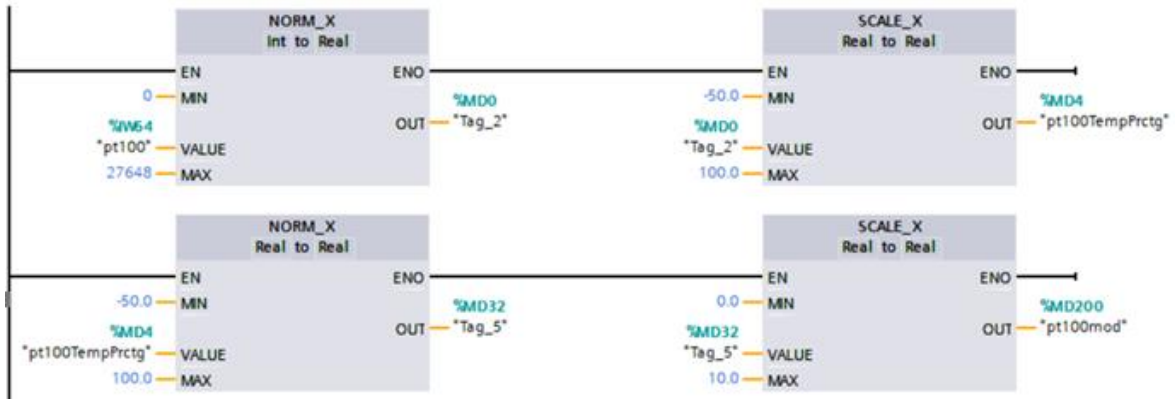
Ladder tempos 20 min et 10 min

Nous avons utilisé deux tempos dont une qui permet d'éteindre la lampe chauffante et toute la platine si celle-ci est utilisée plus de 20 minutes et le second tempo les réactive après 10 minutes d'arrêt.



Ladder blocs soustractions

Les deux ***blocs SUB*** ont pour fonction de soustraire les valeurs de sortie du ***PID compact*** de 3 puis de 100 dans ce cas-ci. Ces valeurs seront stockées dans deux variables de type mémoire qu'on a nommé « SetPointMin » et « OutputPWM ».



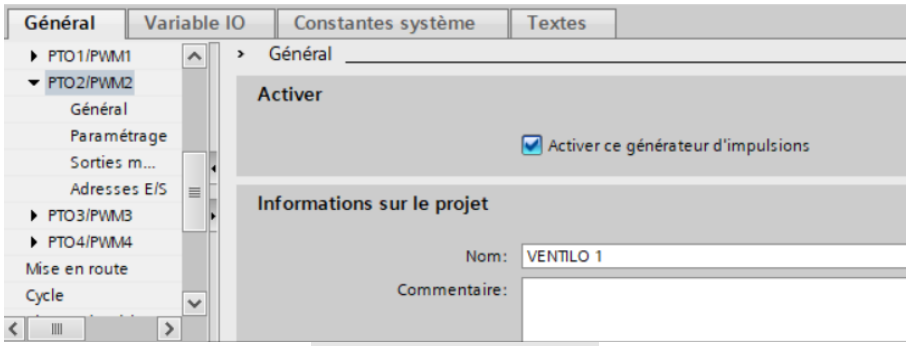
Blocs Norm_X et Scale_X

Nous avons également dû utiliser les fonctions ***Norm_X*** et ***Scale_X*** afin de pouvoir convertir les valeurs analogiques captées à la température de la lampe chauffante. ***Norm_X*** permet de normaliser les valeurs dans une plage spécifique, il ramène un réel entre 0 et 1 tandis que ***Scale_X*** permet de mettre à l'échelle ces valeurs dans une autre plage. On entend par plage une gamme de valeurs dans laquelle une certaine grandeur peut varier. En résumé, ces deux fonctions permettent la mise à l'échelle de valeurs spécifiques.

7. Configuration et programmation du PWM :

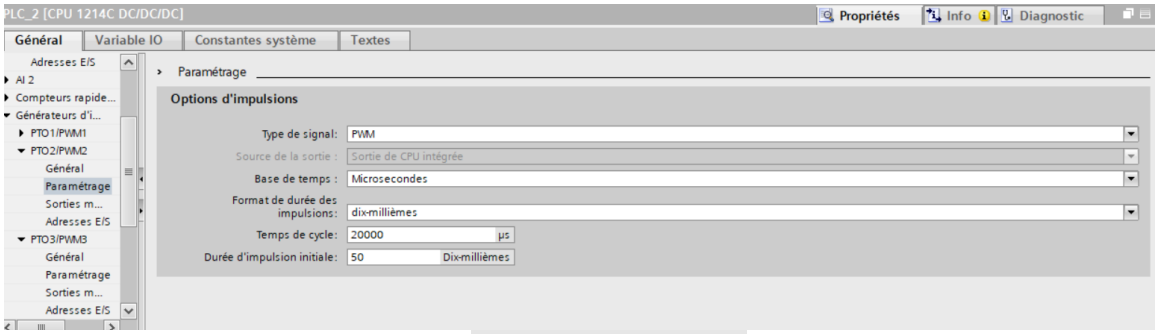
La partie PWM est très importante pour la régulation, on l'utilise pour varier la vitesse des ventilateurs. Le but de notre projet est d'accélérer la vitesse des ventilateurs par rapport à la température du boitier. Par exemple si notre « Setpoint » est à 30° et que la température de notre boitier est à 31°, la vitesse des ventilateurs est censée être à son maximum.

En première partie, il faut qu'on active le générateur de pulsation de certaines sorties
Pour notre cas, l'on a activé le PTO2/PWM2 et le PTO3/PWM3 car nos ventilateurs prennent les adresses Q0.2 et le Q0.4 comme sorties physiques.



Paramètre général PWM

Ensuite, nous devons configurer les options de l'impulsion.
Pour notre projet, nous avions mis la base du temps en microsecondes, le temps de cycle est de 20000 microsecondes et la durée de l'impulsion est de 50 dix-millièmes.

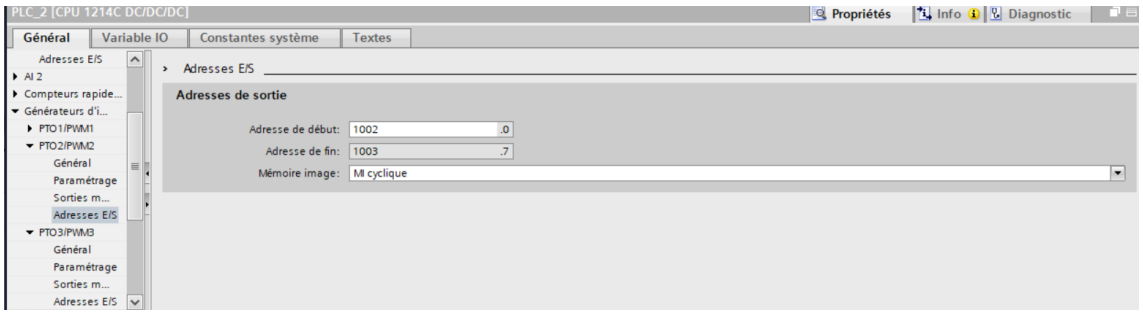


Paramètre d'impulsions

Pour la configuration de la PWM, la sortie matérielle est essentielle, car c'est grâce à cette partie que l'on peut choisir la sortie à laquelle nous allons brancher nos ventilateurs.

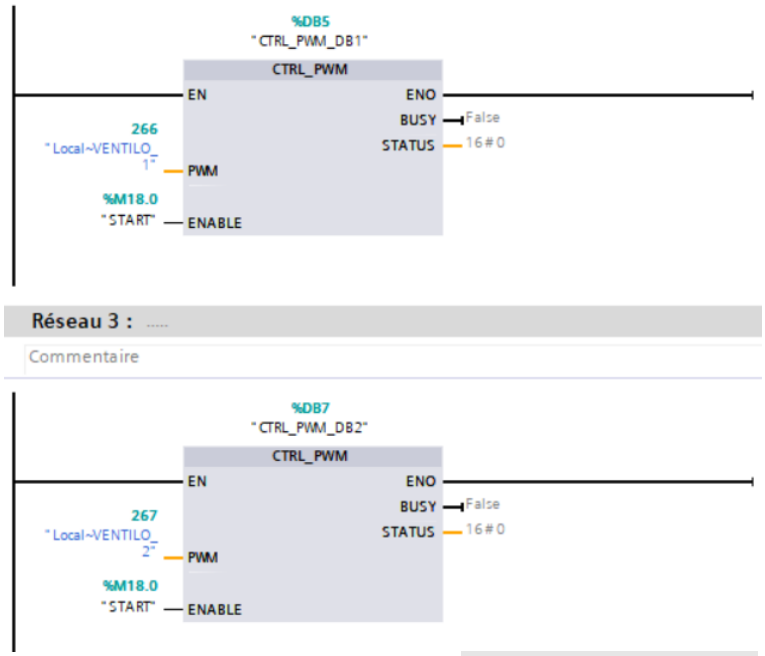
Les sorties matérielles

Pour finir la partie configuration de la PWM, nous devions prendre connaissance de l'adresse de début des sorties. Cette adresse va être réutiliser pour le fonctionnement de la PWM.



Affichage des adresses de sortie

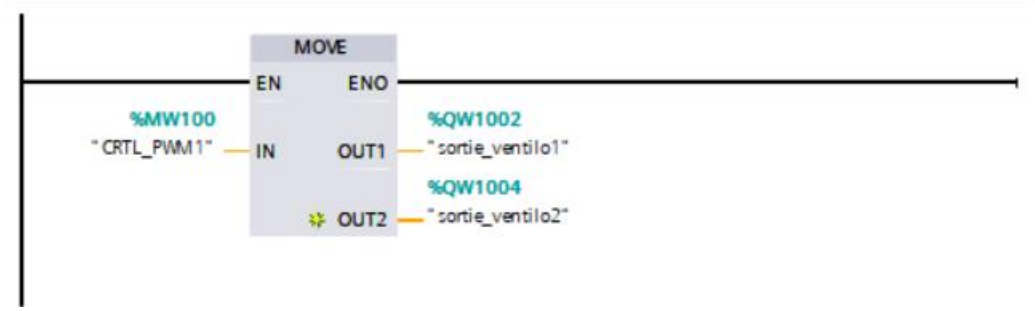
Ensuite, on rajoute un bloc CTRL_PWM (Control-Pulse-With-Modulation). Cette instruction nous permet d'activer et désactiver le générateur d'impulsion à partir du programme.



Le bloc CTRL_PWM_DB

Par la suite, nous utilisons le bloc move. Cette instruction "Copier valeur" nous permet de transférer le contenu de l'opérande à l'entrée IN dans l'opérande à la sortie OUT1.

Pour notre cas, l'on copie la valeur de CRTL_PWM1 dans les deux nouvelles variables qu'on a nommées « sortie_ventilo1 » et « sortie_ventilo2 ». Il faut faire bien attention à choisir l'adresse de début (vu une page au-dessus) lorsqu'on définit les variables .



Bloc move copie vers les sortie PWM

Pour calibrer la vitesse des ventilateurs, on utilise les blocs Norm_X et Scale_X. Si nous avons 27640 comme valeur, les vitesses des ventilateurs seront à leurs maximums.

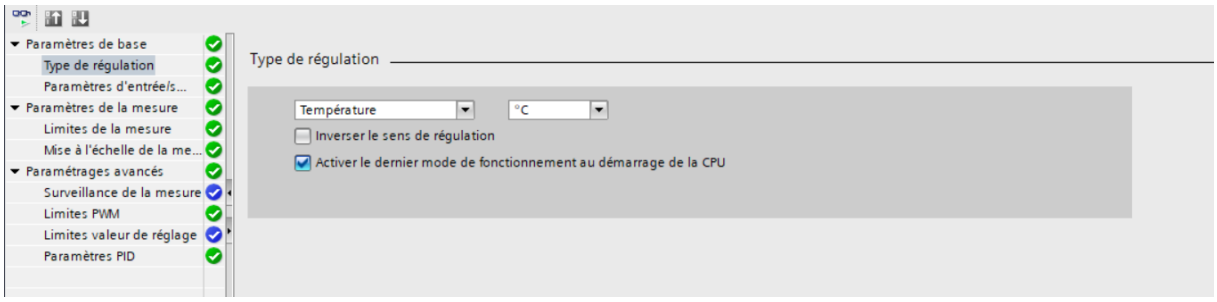
Norm_x ,Scale_x vitesse des ventilateurs

Pour avoir la bonne valeur affichée dans notre HMI, nous avons dû utiliser un bloc de soustraction.

Soustraction valeur PID compact

Dans ce bloc, on soustrait de 100 la valeur que la sortie du PID compact nous envoie et puis on envoie la valeur obtenue dans une variable « OutputPWM ». Nous avons utilisé ce bloc, car la valeur initiale envoyée par le PID compact était inversée.

Nous avions essayé une autre méthode avant celle-ci, nous devions juste cocher la case « Inverser le sens de régulation », mais cela n'a pas fonctionné chez nous.



PID compact inversion de sens

8. Régulation de la température.

La régulation de température dans notre cas particulier est réalisée à l'aide de ventilateurs contrôlés par modulation de largeur d'impulsion (PWM). Cette méthode permet de faire varier la vitesse des ventilateurs de manière proportionnelle à la température mesurée. Les ventilateurs sont utilisés pour maintenir la température de consigne dans un boîtier contenant une lampe chauffante.

Le régulateur PID compare en temps réel la température mesurée à la température de consigne prédéfinie. En fonction de l'écart entre ces deux valeurs, le régulateur ajuste la commande de sortie PWM qui contrôle la vitesse des ventilateurs. L'action proportionnelle assure que plus l'écart entre la température mesurée et la température de consigne est grand, plus la vitesse des ventilateurs sera élevée. L'action intégrale tient compte de l'accumulation de l'écart au fil du temps pour ajuster progressivement la vitesse des ventilateurs. L'action dérivée anticipe les changements rapides de la température et ajuste la commande en conséquence.

Ainsi, grâce à la régulation de température basée sur le régulateur PID et l'utilisation du PWM pour les ventilateurs, nous pouvons maintenir la température de manière précise et stable dans le boîtier. Les ventilateurs s'ajustent en temps réel en fonction de la température mesurée, permettant ainsi de compenser les variations thermiques et d'assurer un contrôle optimal de la température de consigne.

8.1. PID Compact

Pour mettre en œuvre ce principe de régulation PID, nous avons utilisé le bloc PID Compact sur TIA Portal v17.

Le bloc PID (proportionnel-intégral-dérivatif) est un outil de régulation utilisé pour maintenir une variable de processus ou d'une grandeur (par exemple, **la température**, la pression, le débit, etc.) à une valeur de consigne souhaitée. Il calcule la variable de sortie en fonction de l'écart entre la valeur de consigne (Setpoint) et la valeur mesurée, afin de corriger et de stabiliser le processus.

Le bloc PID Compact est une implémentation prête à l'emploi d'un régulateur PID dans TIA Portal. Il est conçu pour simplifier la configuration et la programmation du régulateur PID, en offrant une interface conviviale pour les utilisateurs.

Avec le bloc PID Compact, nous pouvons spécifier et établir les paramètres PID tels que les coefficients proportionnel, intégral et dérivatif, ainsi que les limites de sortie du régulateur.

Une fois que le bloc PID Compact est configuré et intégré dans notre programme d'automatisation, il effectue les calculs nécessaires en temps réel pour générer la variable de sortie appropriée. Cette variable peut être utilisée pour contrôler un actionneur (par exemple, une vanne, un moteur) ou des ventilateurs dans notre projet, afin d'ajuster le processus et de maintenir la variable de processus à la valeur de consigne.

En résumé, le bloc PID Compact dans TIA Portal V17 est un bloc qui facilite la mise en œuvre d'un régulateur PID dans notre projet d'automate industrielle, en nous offrant une solution prête à l'emploi pour la régulation de la température.

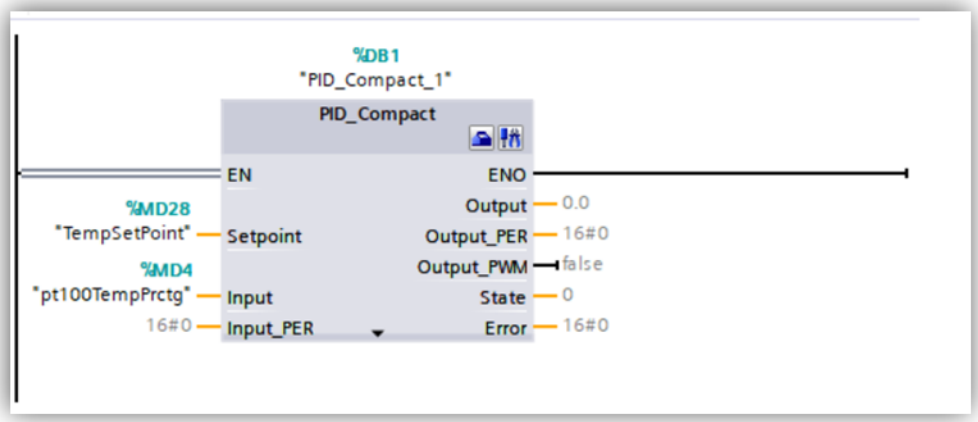
8.1.1. Explication des variables utilisées

Un bloc PID Compact comporte plusieurs variables, mais nous ne les avons pas toutes utilisées car elles n'ont pas été utiles pour notre régulation.

1. Nous avons utilisé, premièrement, la variable ***Setpoint*** qui va nous permettre d'affecter à une variable, la consigne.

2. Nous avons, ensuite, la variable ***Input*** qui va nous permettre d'introduire la valeur de notre température.

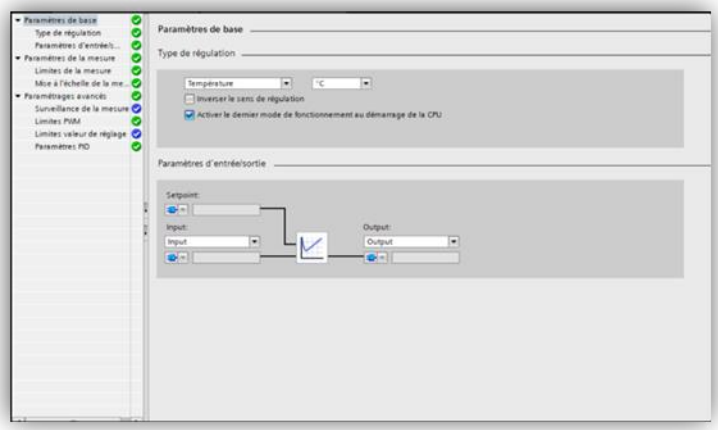
3. Et enfin, il y a la variable ***Output*** qui va nous permettre de récupérer la vitesse de nos ventilateurs, en pourcentages.



Bloc PID_Compact

8.1.2. Explication des fenêtres de configuration

- La première fenêtre, est la fenêtre de configuration de base. Nous allons juste y introduire le type de régulation, si c'est pour de la température, niveau, débit, etc. Nous allons également y introduire nos paramètres d'entrée/sortie.

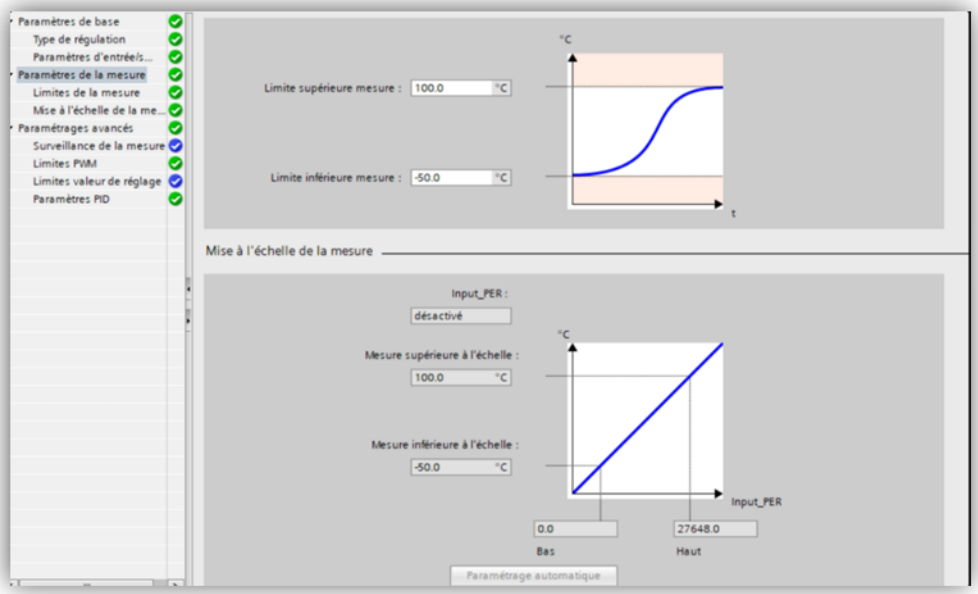


Paramètre de base

- L'onglet 'Paramètre de la mesure' du PID Compact comprend deux sous-onglets importants : ***'Limites de la mesure'*** et ***'Mise à l'échelle de la mesure'***.

- ***'Limites de la mesure'*** permet de définir les valeurs minimale et maximale acceptables pour la mesure de la variable de processus. Cela permet de s'assurer que les mesures restent dans une plage valide, évitant ainsi des valeurs aberrantes ou des erreurs de calcul.

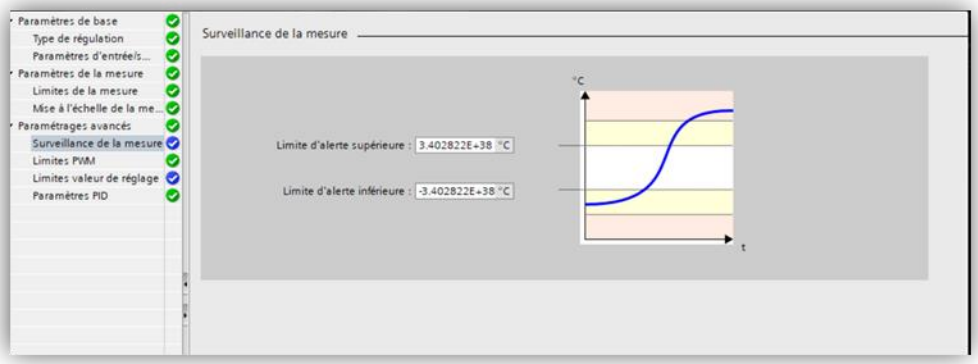
- ***'Mise à l'échelle de la mesure'*** permet d'ajuster les valeurs de la mesure en fonction de la plage de travail souhaitée. C'est tout simplement un genre de bloc ***NormX ScaleX***, mais pour cela il faut utiliser la variable ***Input PER.***



Paramètre de la mesure 1

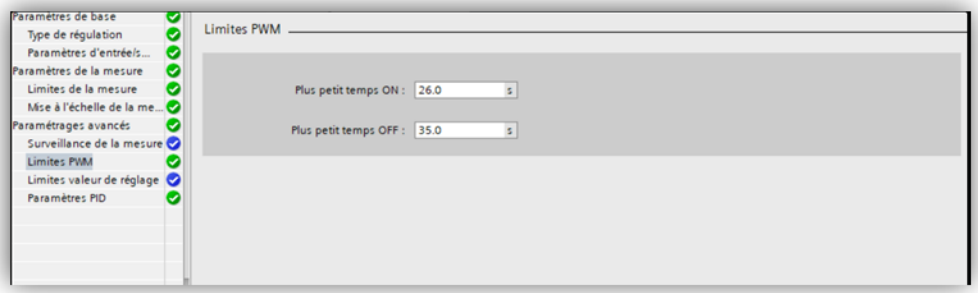
- L'onglet 'Paramètre avancés' du PID Compact comprend quatre sous-onglets importants : '***Surveillance de la mesure***', ***'Limites PWM'***, ***'Limites valeur de réglage'***, ***'Paramètres PID'***.

  - ***'Surveillance de la mesure'*** permet de spécifier des limites d'avertissement pour lesquelles un bit d'avertissement est activé si elles sont dépassées ou si elles sont dépassées. Un bit d'avertissement est activé si elles sont dépassées ou ne sont pas atteints.



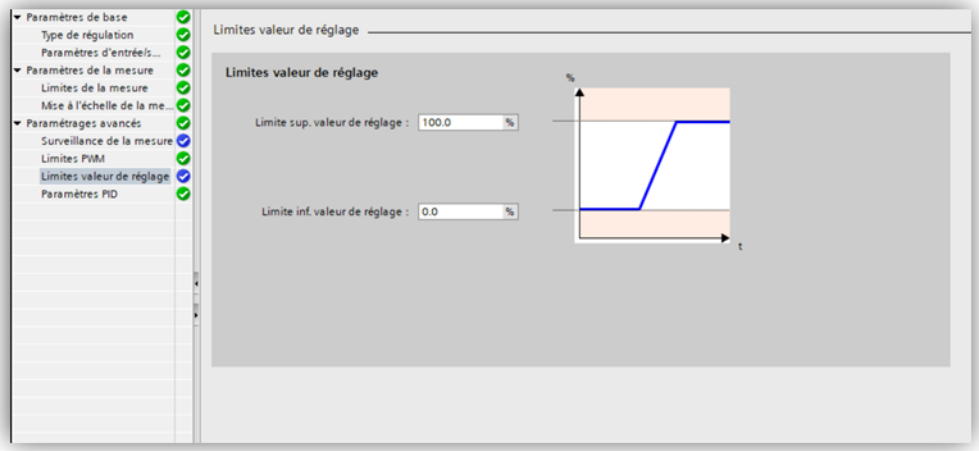
Fenêtre : Surveillance de la mesure

- ***'Limites PWM'*** permet de définir les limites supérieures et inférieures pour la commande de sortie PWM qui contrôle les ventilateurs ou les actionneurs. Cela garantit que la sortie PWM reste dans une plage valide, évitant les valeurs excessives ou nulles qui pourraient affecter la régulation de manière indésirable.



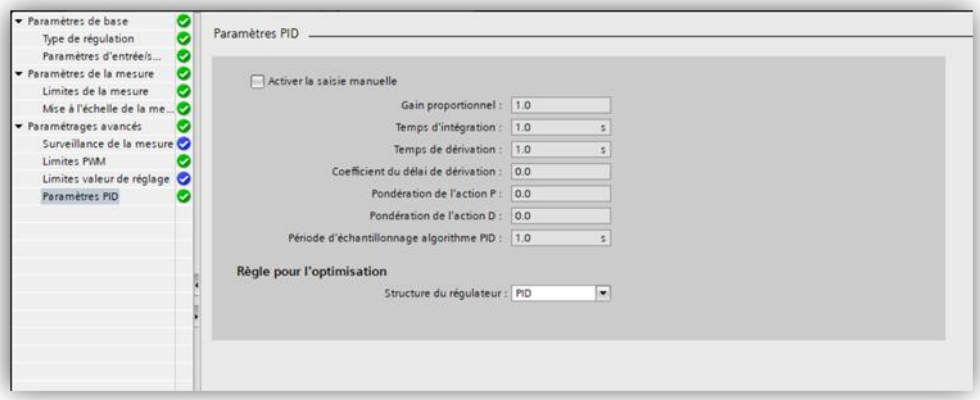
Fenêtre : Limite PWM

- ***'Limites valeur de réglage'*** permet de spécifier les limites supérieures et inférieures pour la valeur de consigne (Setpoint). Ces limites définissent la plage dans laquelle la valeur de consigne peut être ajustée, permettant ainsi un contrôle précis du processus.



Fenêtre : Limite valeur de réglage

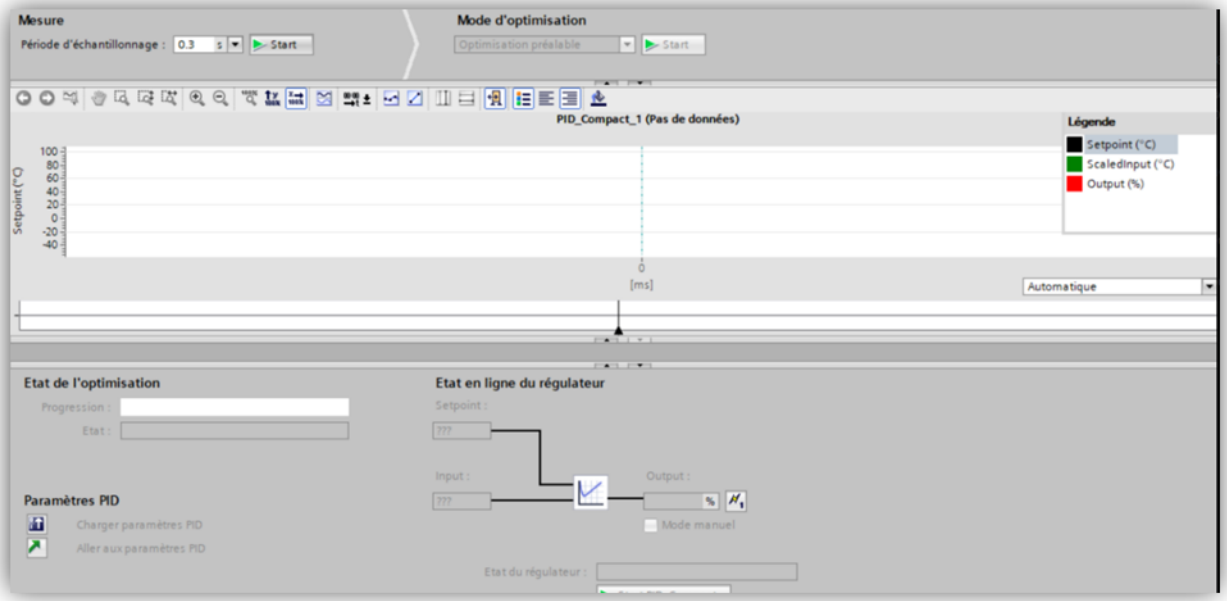
- ***'Paramètres PID'*** regroupe les coefficients proportionnel, intégral et dérivatif du régulateur PID. Ces paramètres peuvent être ajustés pour optimiser la performance de la régulation en fonction des caractéristiques spécifiques du système.



Fenêtre : Paramètre PID

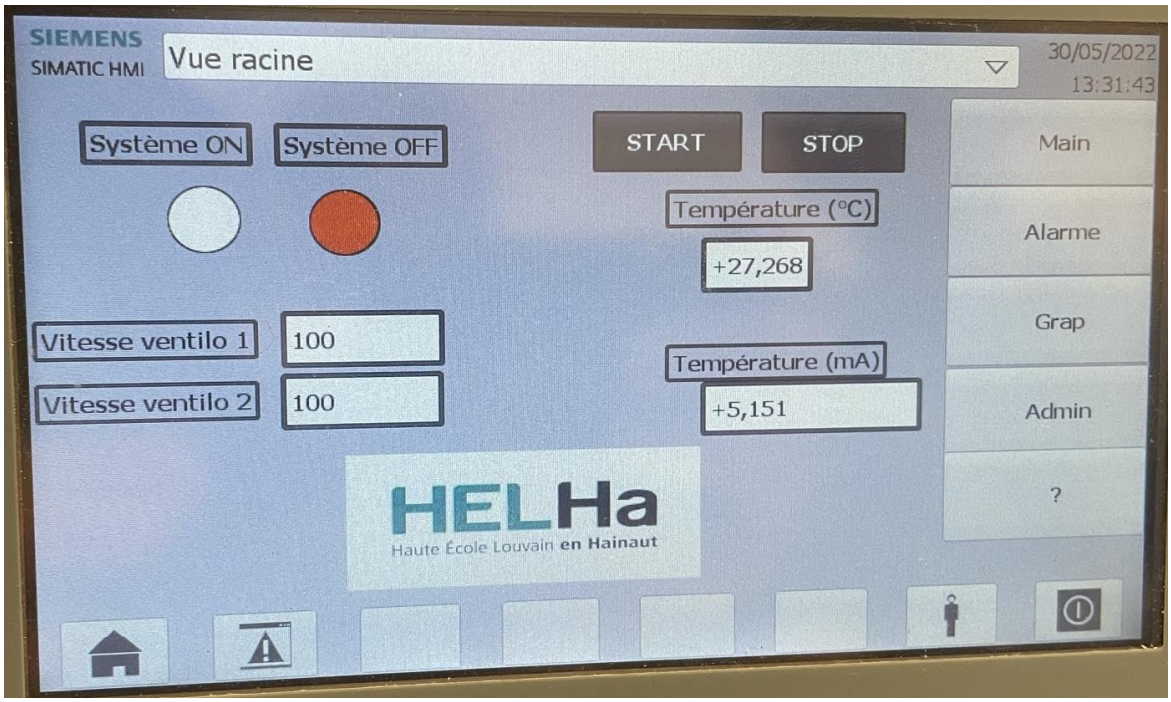
8.1.3. Explication de la fenêtre '*Mise en Service*'

- Cette fenêtre est une interface de l'état de service de notre régulateur, nous pouvons ajuster la période d'échantillonnage et ensuite, en appuyant sur le bouton ***Start,*** on peut démarrer notre régulateur PID. On peut voir le rendu via un graphique tout fait, où il nous montre le ***Setpoint, Scaledinput, Output.*** Le ***Scaledinput*** représente la température en temps réel dans notre boitier et l'***Output*** représente la vitesse des ventilateurs en %.



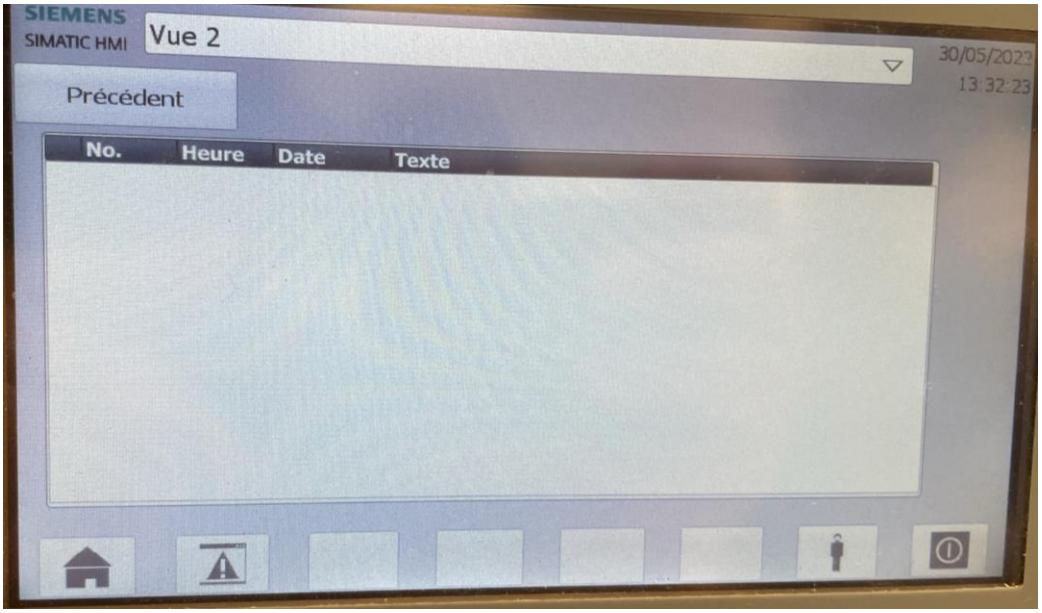
Fenêtre : Mise En Service

9. HMI

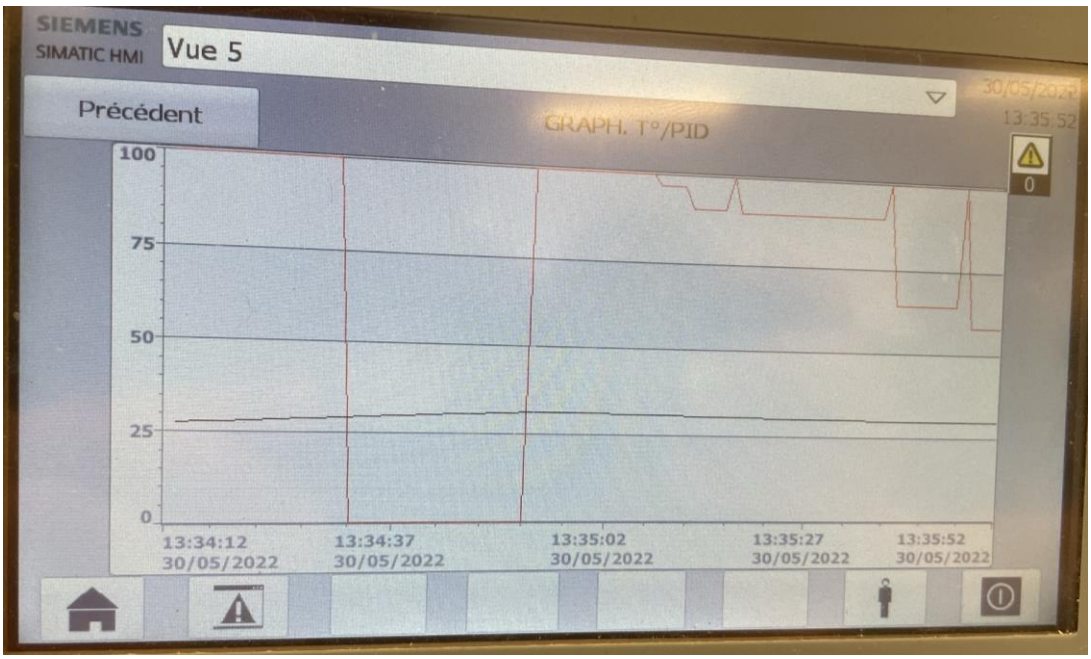


Voici la partie Main, c'est la partie qui apparait par défaut au démarrage.

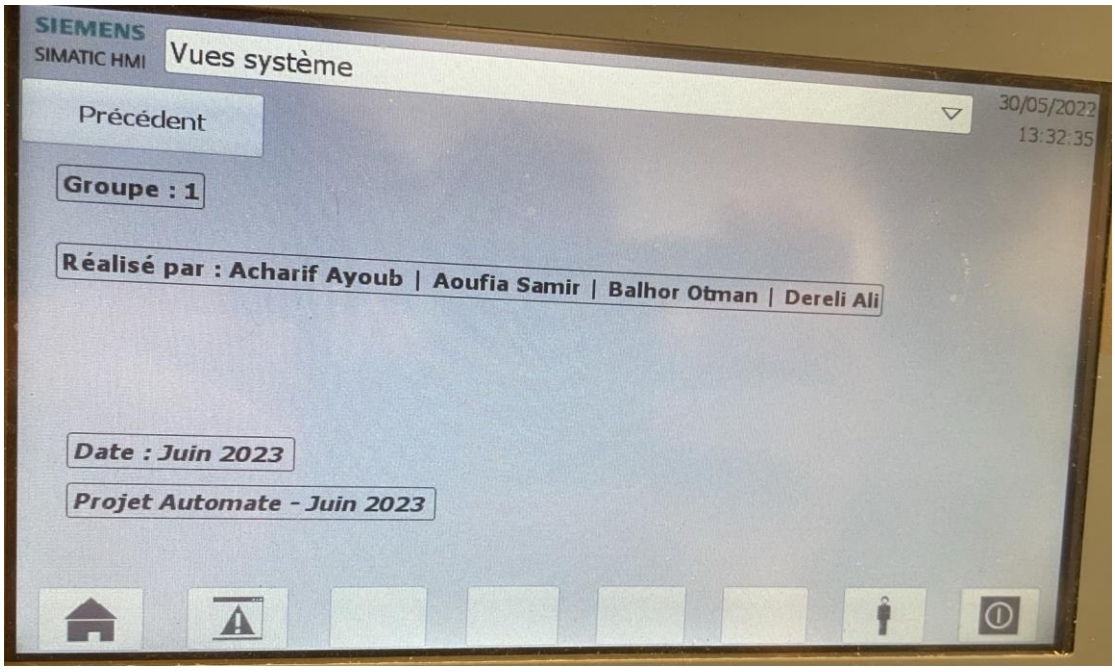
Dans cette partie on retrouve les boutons START / STOP, un système qui indique si le processus tourne grâce à des ronds de couleurs, il y a aussi la température en °C et en mA ainsi que la vitesse des ventilateurs.



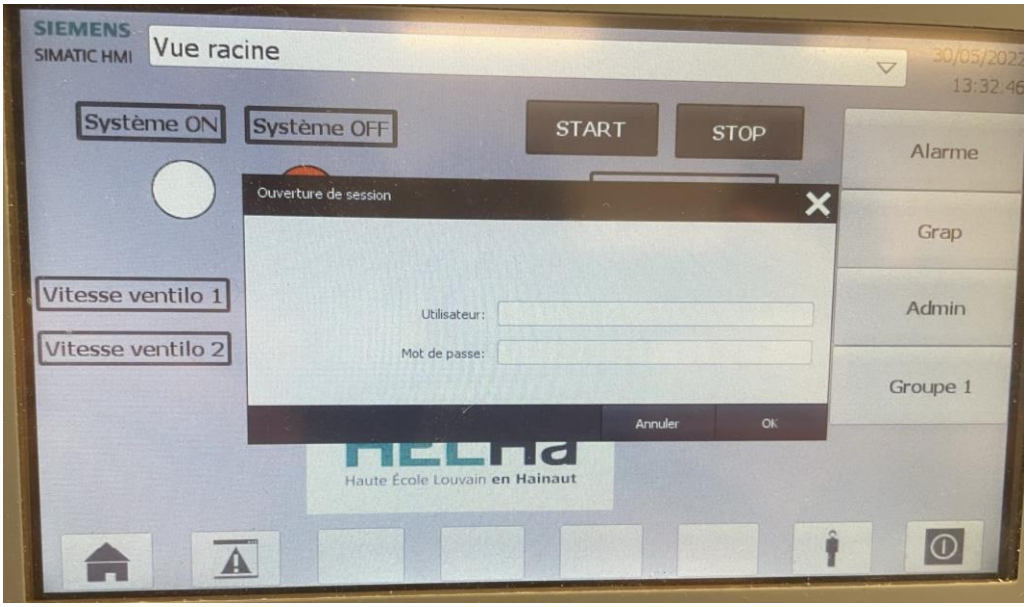
Voici la partie alarme, qui permet l'affichage et la gestion des erreurs. Cette partie sert surtout pour le dépassement de la température.



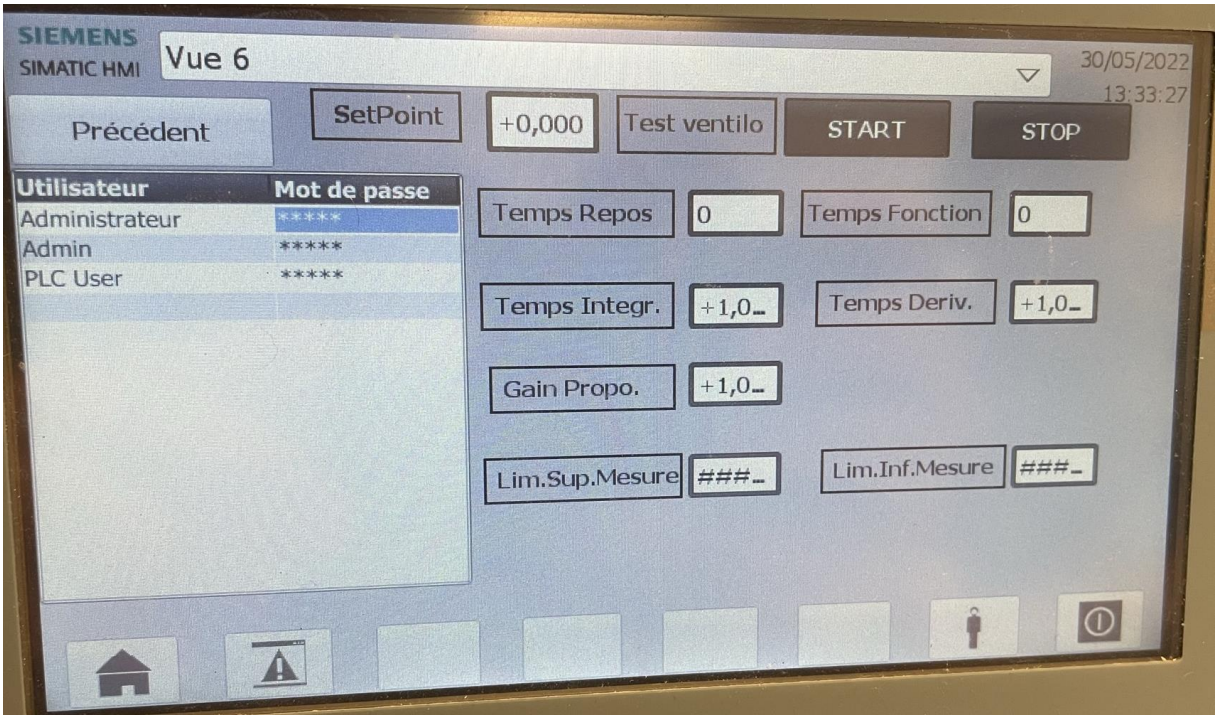
Voici la partie graphique, qui affiche la température par rapport aux temps, il y a aussi une courbe qui montre la température souhaitée.



Voici la partie contact, qui affiche le numéro de groupe ainsi que les personnes du groupe, la date et le nom du cours.



Voici la partie admin pour y accéder nous devons rentrer un utilisateur (admin) ainsi que le mot de passe (admin).



Une fois rentrer dans la partie admin, nous avons accès a 2 boutons START / STOP qui permet de tester les ventilateurs en les faisant tourner seul. On doit mettre un set point (la température souhaité), il y a aussi la modification des mots de passe utilisateur.

On peut choisir combien de temps le programme doit fonctionner (temps fonction) avant d'activer le temps de repos. On a le moyen de changer les valeurs PID.

On a mis le moyen changer les limites de la mesure mais cela ne fonctionne pas car il faut utiliser la partie input PER.

10. PUT & GET

Nous devions faire communiquer l'automate 1200 avec l'automate 1500 mais malheureusement nous n'avons pas eu le temps pour faire cela.

PUT & GET sont des blocs qui permettent la communication entre 2 automates.

L'instruction GET est utilisée pour demander, récupérer et lire des informations, un automate fait une demande à un autre automate pour avoir certaines informations, Ensuite grâce à l'instruction PUT on envoie les informations demandées par le GET.

11. Conclusion

En conclusion, notre travail de groupe avait pour objectif d'établir un système automatisé de régulation et de supervision à distance. Nous avons réussi à progresser, en premier lieu, en réalisant le montage électrique du matériel requis.

Par la suite, nous avons mis en place un système de régulation en utilisant le PLC-1214, le boîtier de régulation contenant une PT100 et une lampe chauffante. Et enfin, nous sommes parvenus à faire une gestion d'évènements, et une supervision avec un HMI. Ce projet nous a offert une occasion précieuse de mettre en pratique notre capacité à travailler en équipe dans le domaine des automates industriels et de la programmation.

En résumé, cette expérience nous a permis de développer nos compétences pratiques et d'approfondir notre compréhension dans un environnement concret.