REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHECHE SCIENTIFIQUE

Université Mohamed Seddik Benyahia – Jijel

Faculté des Sciences et de la Technologie
Département d'Automatique

PROJET DE FIN D'ETUDE POUR L'OBTENTION DU DIPLOME
MASTER EN AUTOMATIQUE

OPTION : Automatique et Système

Thème

Régulation PID par API et supervision du niveau d'un réservoir cylindrique

Réalisé par :

- ***Aliliche Loqman***
- ***Messadi Abdelkrim***

Encadré par :

Pr. Labiod Salim

Promotion 2020

Résumé :

Les automates programmables industriels représentent l'élément important de la chaîne automatisée, car ils assurent une meilleure flexibilité et facilitent la maintenance. Destiner à la conduite et la surveillance en temps réel de processus industriels

Le but de ce travail est de régulé le niveau d'un réservoir de stockage cylindrique assurée par un régulateur PID intégré comme bloc programmé dans un API S7/300 dans le logiciel Step7 de siemens et simulé sur PLCSIM (CPU virtuelle) avec une visualisation réalisée par une interface homme-machine à base du logiciel WinCC flexible, celle-ci permet entre autres, la récupération et l'affichage en ligne des variables caractéristique du processus. De plus, ce travail permet de présenter l'utilité des logiciels de simulation des systèmes automatisés pour l'acquisition de nouvelles compétences dans le domaine du contrôle/commande des systèmes industriels. A la fin, les résultats de simulation sont présentés et commentés.

Mots clé : Automates programmable industriel, Régulateur PID, Supervision, SIMATIC, winCC flexible, factory I/O

Abstract

Industrial programmable logic controllers are an important part of the automated chain, as they provide greater flexibility and facilitate maintenance. Intended for the real-time control and monitoring of industrial processes.

The aim of this work is to regulate the level of a cylindrical storage tank ensured by a PID regulator integrated as a block programmed in an S7 / 300 PLC in the Step7 software from siemens and simulated on PLCSIM (virtual CPU) with a visualization carried out Via a human-machine interface based on the WinCC flexible software, this enables, among other things, the online retrieval and display of process-characteristic variables. In addition, this work makes it possible to present the utility of simulation software for automated systems for the acquisition of new skills in the field of control / command of industrial systems. At the end, the simulation results are presented and commented on.

Key words : Industrial programmable logic controllers, PID regulator, Supervision, SIMATIC, winCC flexible, factory I / O

ملخص

تعد وحدات التحكم المنطقية القابلة للبرمجة الصناعية جزءًا مهمًا من السلسلة الآلية لأنها توفر قدرًا أكبر من المرونة وتسهل الصيانة. مخصص للتحكم في الوقت الحقيقي ومراقبة العمليات الصناعية

الهدف من هذا العمل هو تنظيم مستوى خزان التخزين الاسطواني الذي يضمنه منظم PIDمدمج في S7/300 PLC في برنامج STEP 7 من SIEMENS ومحاكاة على PLCSIM (وحدة المعالجة المركزية الافتراضية) مع تصوير تم تنفيذه من خلال واجهة بين الانسان والالة تعتمد على برنامج winCC flexible، يتيح ذلك، من بين أشياء أخرى، الاسترداد المباشر وعرض متغيرات وخصائص العملية بالإضافة الى ذلك، يتيح هذا العمل إمكانية تقديم فائدة برامج المحاكاة للأنظمة الالية لاكتساب مهارات جديدة في مجال التحكم في الأنظمة الصناعية. في النهاية، يتم عرض نتائج المحاكات والتعليق عنها.

كلمات مفاتيح: وحدات التحكم المنطقية القابلة للبرمجة الصناعية، منظم PID، الاشراف، winCC flexible، SIMATIC، factory I/O

Remerciements

Avant tout nous remercions Dieu Le tout puissant de nous avoir donné le courage, la volonté, la patience, et la santé durant toutes ces années et que grâce à lui ce travail a pu être réalisé.

Ainsi, nous tenons également à exprimer nos vifs remerciements à notre encadreur M. Salim Labiod pour avoir d'abord proposé ce thème, pour le suivi continuel tout le long de la réalisation de ce mémoire, et qui n'a pas cessé de nous donner des conseils et des remarques.

Nos remerciements vont aussi à tous les enseignants et le chef de département d'Automatique qui ont contribué à notre formation, et à tous les membres du jury qui ont accepté de juger notre travail.

Enfin nous tenons à exprimer notre reconnaissance à tous nos amis et collègues pour leur soutien et leur encouragement afin de terminer ce travail.

Je dédié ce projet :

A ma chère mère,
A mon cher père,
Qui n'ont jamais cessé, de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs
A mes sœurs,
Pour ses soutiens moraux et leurs conseils précieux tout au long de mes études
A mes oncles et mes tantes,
A mes cousin et cousines,
A mon cher binôme 'LOQMAN',
Pour son entente et son sympathie
A Tous mes amis,
A tous ceux qui m'ont aidé de près ou loin pour la réalisation de ce modeste travail,
Tous mes camarades de la promotion Master II
et tous les autres sans exception.

A.Krim

Je dédié ce projet :

A ma chère mère,
A mon cher père,
Qui n'ont jamais cessé, de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs
A mes frères, A mes sœurs,
Pour ses soutiens moraux et leurs conseils précieux tout au long de mes études
A mes oncles et mes tantes,
A mes cousin et cousines,
A mon cher binôme 'A.KRIM',
Pour son entente et son sympathie
A Tous mes amis,
A tous ceux qui m'ont aidé de près ou loin pour la réalisation de ce modeste travail,
Tous mes camarades de la promotion Master II
et tous les autres sans exception.

Lokman

Table des matières

Chapitre 3 : Automates programmables industriels

Chapitre 4 : Régulation et supervision du niveau d'un réservoir cylindrique

Table des figures

Liste des tableaux

Table des Symboles

Notations

q_{in} Flux d'entrée

q_{out} Flux de sortie

$V(t)$ Le volume de liquide à l'intérieur du réservoir

k_v Le coefficient de la vanne

$h(t)$ Niveau de liquide dans le réservoir

$\dot{X} = \frac{dx(t)}{dt}$ La dérivée de la variable x par rapport au temps

K_p Gain proportionnel

T_i La constante de temps intégrale

T_d La constante de temps dérivée

T_e Le temps de discrétisation

Variables

y Variable de sortie

x Variable d'état

t Variable temps

z Variable dans domaine échantillonné

u Variable manipulée (Commande)

e Erreur

∂ Dérivée partielle

Introduction Générale

Introduction générale

La régulation de niveau des liquides dans les systèmes à réservoirs est un problème classique de la régulation industrielle. En général, de nombreuses applications industrielles sont concernées par la régulation de niveau tel que la colonne à distillation, les chaudières, et les raffineries de pétrole dans les industries pétrochimiques. En fait, les industries de transformation telles que les industries pétrochimiques, la fabrication du papier et le traitement des eaux exigent que les liquides soient pompés, stockés dans des réservoirs. Ainsi, la régulation du niveau du liquide dans les réservoirs est un problème fondamental dans les industries de transformation [1]. Cette régulation est généralement assurée par des régulateurs PID analogiques ou des régulateurs PID intégrés dans des automates programmables industriels.

L'objectif de ce travail est de régulé le niveau d'un réservoir de stockage cylindrique simulé dans le logiciel Factory I/O. Cette régulation est assurée par un régulateur PID intégré comme bloc programmé dans un API S7/300 dans le logiciel Step7 de siemens et simulé sur PLCSIM (CPU virtuelle) avec une visualisation réalisée par le logiciel winCC flexible. De plus, ce travail permet de présenter l'utilité des logiciels de simulation des systèmes automatisés pour l'acquisition de nouvelles compétences dans le domaine du contrôle/commande des systèmes industriels.

Ce mémoire est organisé en quatre chapitres :

Le premier chapitre présente la description et la modélisation des réservoirs conique, sphérique, cylindrique horizontal et cylindrique vertical. Aussi, des résultats de simulation en boucle ouverte sont présentés et commentés.

Le deuxième chapitre présente brièvement les notions de la régulation automatique et les différentes formes des régulateurs PID avec des méthodes simples pour le calcul des paramètres de ces régulateurs. A la fin du chapitre, différentes approches de discrétisation des régulateurs PID analogiques sont données.

Le troisième chapitre présente en particulier les automates programmables industriels (API) de siemens, les régulateurs PID intégrés ainsi que le logiciel de programmation des automates Step 7 et le logiciel WinCC flexible 2008 qui est utilisé pour les interfaces homme-machine (IHM).

Le dernier chapitre présente le système à réservoir de stockage cylindrique de Factory I/O et les différentes implémentations de la régulation PID dans Step 7 avec la partie IHM. A la fin du chapitre, des résultats de simulation sont présentés et commentés.

CHAPITRE 1

Modélisation mathématique

1.1 Introduction

Pour répondre à la grande variété des produits liquides industrielles à stocker, les constructeurs ont recours à des réservoirs de formes diverses et de conceptions différentes. On trouve essentiellement les réservoirs conique, sphérique, cylindrique horizontal et cylindrique vertical, conçus pour s'adapter le plus rationnellement et le plus économiquement possible aux caractéristiques du produit à traiter [1].

La configuration d'un réservoir dépend de deux facteurs essentiels qui sont, d'une part, la tenue de la structure à la pression interne développée par le produit, et d'autre part le maintien du liquide à un certain niveau pour faciliter l'exploitation.

La forme cylindrique est la plus courante en raison de sa simplicité de mise en œuvre et de sa bonne résistance à la pression interne. Lorsque la pression interne est importante, on a recours à des formes sphériques mieux adaptées que les cylindres et qui permettent de réduire les épaisseurs des parois [1].

Ce chapitre présente les modèles mathématiques non linéaires des réservoirs conique, sphérique, cylindrique horizontal et cylindrique vertical en temps continu et discret. A la fin du chapitre, des résultats de simulation sont présentés et commentés.

1.2 Modélisation mathématique d'un réservoir

Le modèle mathématique dynamique d'un réservoir avec un débit d'entrée noté $q_{in}(t)$ et un débit de sortie donné par $q_{out}(t)$ est donné par une équation du bilan massique de la forme suivante :

$$q_{in}(t) = q_{out}(t) + \frac{dV(t)}{dt} \tag{1.1}$$

Le volume du liquide à l'intérieur du réservoir V(t) est donné par la relation :

$$V(t) = f(F(h(t))) \tag{1.2}$$

Où $F(h(t))$ représente la base du réservoir qui dépend du niveau du liquide $h(t)$ dans le réservoir. En outre, il découle du principe de Bernoulli que le débit de sortie en fonction du niveau du liquide dans le réservoir peut être exprimé comme suit :

$$q_{out}(t) = k_v\sqrt{h(t)} \tag{1.3}$$

Où k_v représente le coefficient de la vanne. A partir des équations (1.2) et (1.3), nous pouvons réécrire le modèle (1.1) sous la forme :

$$q_{in}(t) = k_v\sqrt{h(t)} + \frac{dV(t)}{dh}\frac{dh(t)}{dt} \tag{1.4}$$

Après la substitution de $\frac{dV(t)}{dh}$ par *F(h)*, on obtient le modèle mathématique non linéaire suivant :

$$q_{in}(t) = k_v\sqrt{h(t)} + F(h)\frac{dh(t)}{dt} \tag{1.5}$$

En général, le modèle d'état d'un système non linéaire est donné par les relations suivantes :

$$\begin{aligned} \frac{dx(t)}{dt} &= f(x(t), u(t), t) \\ y(t) &= g(x(t), u(t), t) \end{aligned} \tag{1.6}$$

Où *x(t)* est le vecteur d'état, *u(t)* l'entrée de commande, *y(t)* la sortie du système, et f (·) et g (·) sont des fonctions non linéaires.

En réécrivant le modèle dans (1.5) sous la forme du modèle d'état général défini dans (1.6), nous obtenons un modèle mathématique non linéaire général de la forme :

$$\begin{aligned} \frac{dh(t)}{dt} &= \frac{q_{in}(t) - k_v\sqrt{h(t)}}{F(h)} \\ y(t) &= h(t) \end{aligned} \tag{1.7}$$

Il est souvent pratique de reformuler le modèle mathématique en temps continu en temps discret afin de concevoir des contrôleurs discrets avancés. Comme nous avons défini le modèle d'état non linéaire en temps continu dans (1.6), nous pouvons également définir le modèle d'état non linéaire en temps discret comme suit :

$$\begin{aligned} x(t + T_s) &= x(t) + T_s f(x(t), u(t), t) \\ y(t) &= g(x(t), u(t), t) \end{aligned} \tag{1.8}$$

Où T_s est la période d'échantillonnage.

Dans la section suivante nous présentons les modèles dynamiques des réservoirs coniques, sphériques, horizontaux-cylindriques et verticaux-cylindriques.

1.2.1 Réservoir de stockage conique

La représentation géométrique du réservoir conique est donnée sur la Figure 1.1. Le modèle mathématique d'un tel processus est dérivé en exprimant le volume du tronc en fonction du niveau du liquide. Le réservoir de stockage conique est caractérisé par les variables R_1 et R_2 qui sont des rayons de la base inférieure et supérieure, et par la hauteur maximale du réservoir de stockage $h_{\max}$. Le volume du liquide à l'intérieur du tronc est donné par :

$$V_f(h(t)) = \frac{\pi h(t)}{3}(r_f^2(h(t)) + R_2 r_f(h(t)) + R_2^2) \tag{1.9}$$

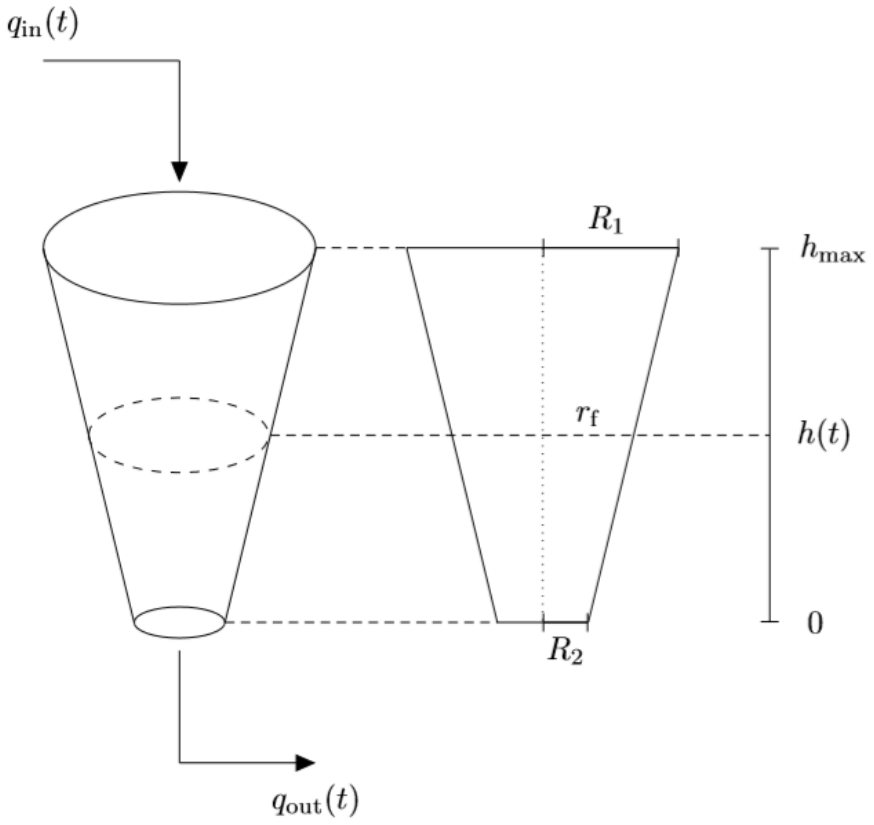


Figure 1.1 : Illustration du réservoir de forme conique

La variable $r_f(h(t))$ représente le rayon d'un disque représentant la surface du liquide au niveau $h(t)$. Le rayon $r_f(h(t))$ est une fonction explicite du niveau de liquide, exprimé par :

$$r_f(h(t)) = R_2 + \frac{R_1 - R_2}{h_{\max}}.h(t) \tag{1.10}$$

En substituant l'expression (1.10) dans (1.9), on obtient :

$$V_f(h(t)) = \frac{\pi h(t)}{3}\left(3R_2^2 + 3R_2 + \frac{R_1 - R_2}{h_{\max}}h(t) + \left(\frac{R_1 - R_2}{h_{\max}}\right)^2 h^2(t)\right). \quad (1.11)$$

Maintenant, nous pouvons combiner l'expression du volume dans (1.11) et le modèle général du bilan massique dans (1.5), ce qui donne :

$$q_{in}(t) = k_v\sqrt{h(t)} + \pi\left(R_2 + h(t)\frac{R_1 - R_2}{h_{\max}}\right)^2 \frac{dh(t)}{dt}. \quad (1.12)$$

Par conséquent, le modèle mathématique dynamique non linéaire du réservoir de stockage conique avec l'équation de sortie peut être exprimé comme suit :

$$\frac{dh(t)}{dt} = \frac{q_{in}(t) - k_v\sqrt{h(t)}}{\pi\left(R_2 + h(t)\frac{R_1 - R_2}{h_{\max}}\right)^2} \quad (1.13a)$$

$$y(t) = h(t) \quad (1.13b)$$

Le modèle mathématique non linéaire du réservoir conique en temps discret est défini à partir de (1.8), comme suit :

$$h(t + T_s) = h(t) + T_s.\frac{q_{in}(t) - k_v\sqrt{h(t)}}{\pi\left(R_2 + h(t)\frac{R_1 - R_2}{h_{\max}}\right)^2} \quad (1.14)$$

1.2.2 Réservoir de stockage sphérique

La représentation schématique du réservoir de stockage sphérique est donnée sur la Figure 1.2. Le modèle mathématique de ce processus est établi en exprimant le volume du segment sphérique d'une base en fonction du niveau de liquide. Ce processus est caractérisé par la variable *R*, qui représente le rayon de la sphère. Le volume du liquide à l'intérieur du réservoir sphérique est donné par :

$$V_s(h(t)) = \frac{\pi h(t)}{6}(3r_s^2(h(t)) + h^2(t)), \quad (1.15)$$

Où la variable $r_s(h(t))$ représente le rayon de la section du réservoir au niveau $h(t)$.

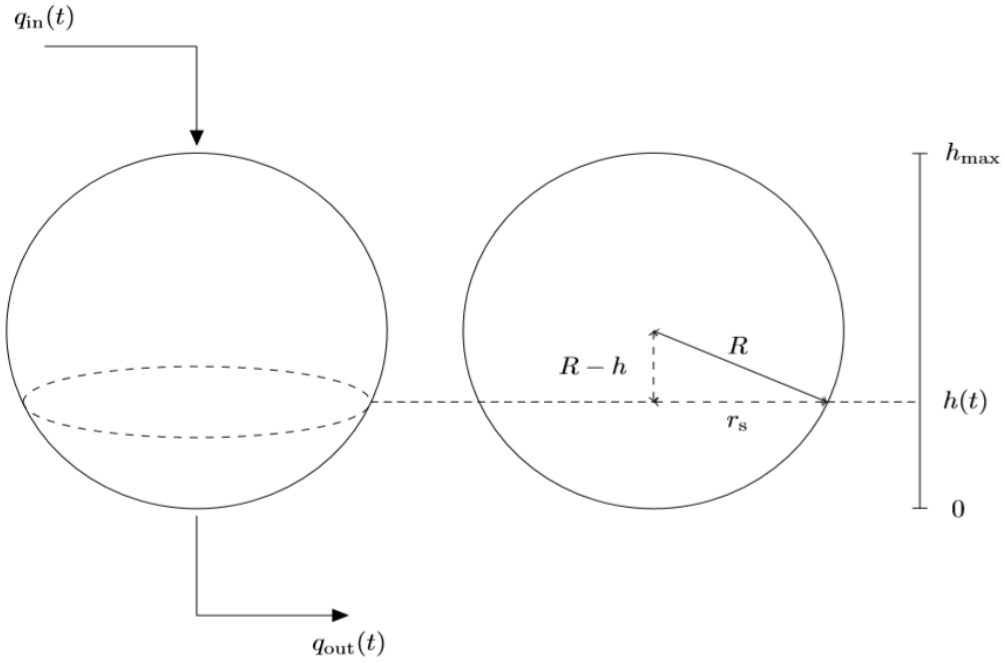


Figure 1.2 : Illustration du réservoir de la forme sphérique

Le rayon en fonction du niveau de liquide est exprimé par :

$$r_s(h(t)) = \sqrt{R^2 - (R - h(t))^2} \tag{1.16}$$

En substituant l'expression (1.16) dans (1.15), on obtient :

$$V_s(h(t)) = \frac{\pi h(t)}{3}(3Rh(t) - h^2(t)). \tag{1.17}$$

Ensuite, nous combinons l'expression du volume dans (1.17) et le modèle général du bilan de masse défini dans (1.5), ce qui donne :

$$q_{in}(t) = k_v\sqrt{h(t)} + \pi(2Rh(t) - h^2(t))\frac{dh(t)}{dt}. \tag{1.18}$$

Ce qui permet d'écrire le modèle d'état non linéaire du réservoir de stockage sphérique :

$$\begin{aligned} &\frac{dh(t)}{dt} = \frac{q_{in}(t) - k_v\sqrt{h(t)}}{\pi(2Rh(t) - h^2(t))}, \\ &y(t) = h(t), \end{aligned} \tag{1.19}$$

Le modèle mathématique non linéaire du réservoir sphérique en temps discret est donné par :

$$h(t + T_s) = h(t) + T_s.\frac{q_{in}(t) - k_v\sqrt{h(t)}}{\pi(2Rh(t) - h^2(t))}. \tag{1.20}$$

1.2.3 Réservoir de stockage cylindrique horizontal

La représentation géométrique du réservoir de stockage cylindrique horizontal est représentée sur la Figure 1.3. Le modèle dynamique est établi en exprimant le volume du segment circulaire en fonction du niveau du liquide.

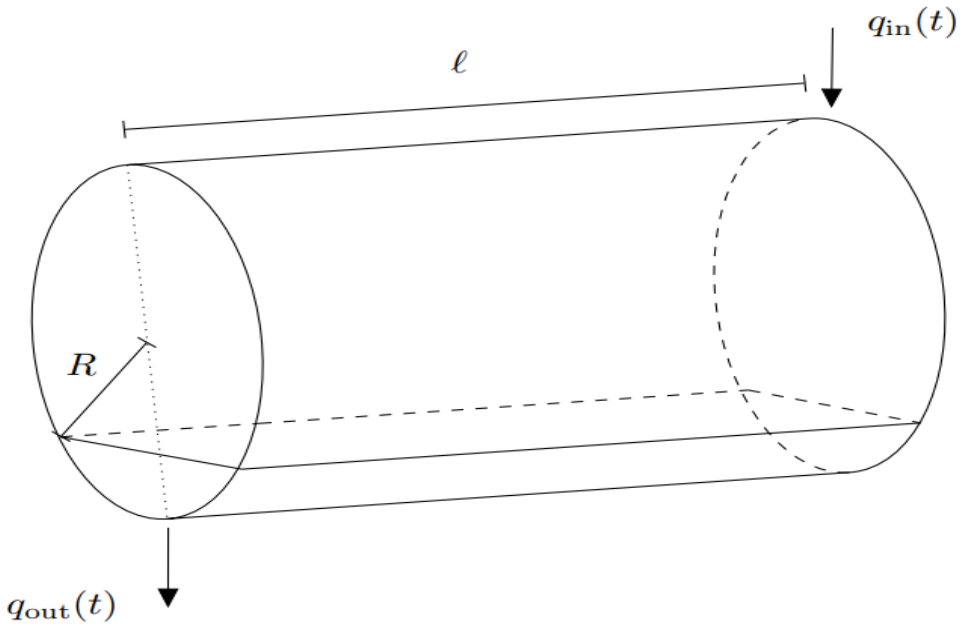


Figure 1.3 : Illustration du réservoir de stockage cylindrique horizontal

Le réservoir est caractérisé par des variables R qui représentent le rayon de la base supérieure et inférieure du réservoir cylindrique et par ℓ qui est la longueur du réservoir cylindrique. Le volume de liquide à l'intérieur du réservoir en fonction de $h(t)$ est donné par :

$$V_c(h(t)) = (R^2 \arccos(\frac{R-h(t)}{R}) - (R-h(t))\sqrt{2Rh(t)-h^2(t)})\ell. \tag{1.21}$$

En combinant l'expression du volume dans (1.21) et le bilan de masse général défini dans (1.5), nous obtenons :

$$q_{in}(t) = k_v\sqrt{h(t)} + 2\ell\sqrt{h(t)(2R)-h(t)}\,\frac{dh(t)}{dt} \tag{1.22}$$

En réécrivant le modèle dans (1.22) sous la forme de modèle d'état général défini dans (1.7), nous obtenons le modèle mathématique non linéaire de réservoir de stockage cylindrique horizontal suivant :

$$\begin{aligned} \frac{dh(t)}{dt} &= \frac{q_{in}(t) - k_v\sqrt{h(t)}}{2\ell\sqrt{h(t)(2R-h(t))}}, \\ y(t) &= h(t), \end{aligned} \tag{1.23}$$

Le modèle mathématique non linéaire du réservoir cylindrique horizontal en temps discret peut être obtenu à partir de (1.23).

1.2.4 Réservoir de stockage cylindrique vertical

Dans cette section, nous nous concentrerons sur le réservoir de stockage cylindrique vertical qui ressemble au réservoir de stockage cylindrique horizontal. La représentation géométrique du réservoir de stockage cylindrique vertical est représentée sur la Figure 1.4. Le modèle d'un tel processus est dérivé comme le réservoir précédent, en exprimant le volume du segment circulaire en fonction du niveau du liquide.

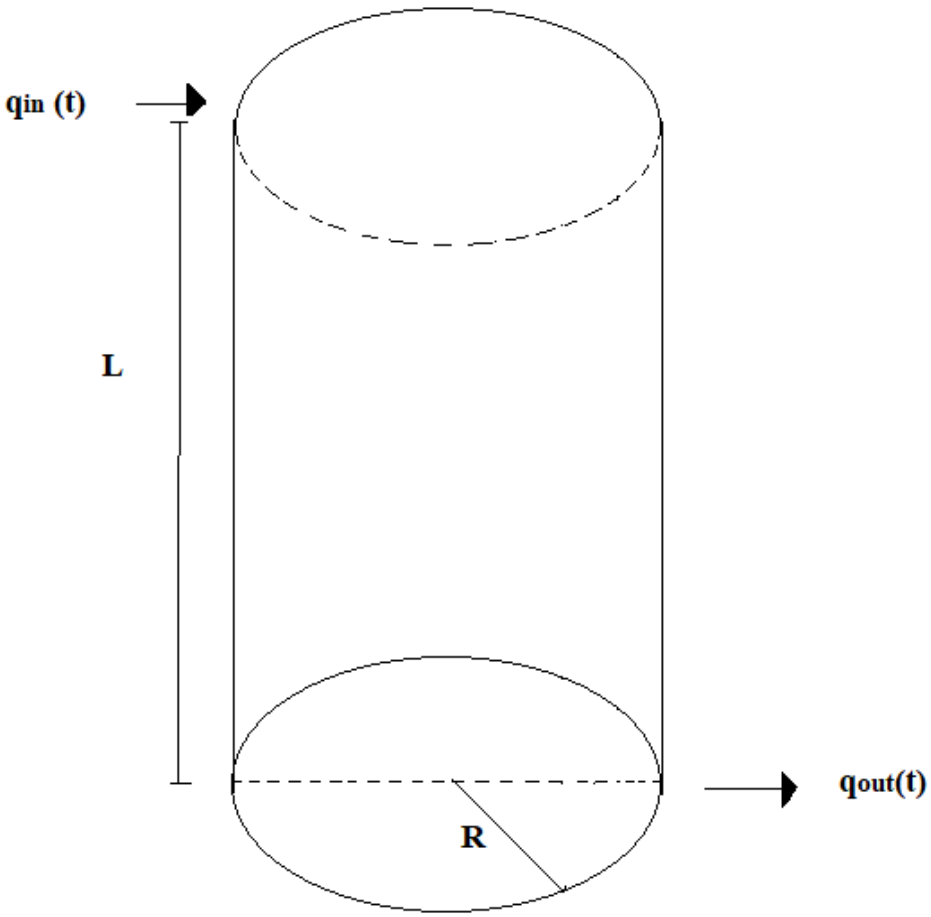


Figure 1.4 : Illustration du réservoir de stockage cylindrique vertical

Le réservoir est caractérisé par des variables *R* qui représentent le rayon de la base supérieure et inférieure du réservoir cylindrique et par ℓ qui est la longueur du réservoir cylindrique. Le volume de liquide à l'intérieur du réservoir en fonction de $h(t)$ est donné par :

$$v_c(h(t)) = \pi R^2 h(t) \tag{1.24}$$

En combinant l'expression du volume dans (1.24) et le bilan de masse général défini dans (1.5), nous obtenons :

$$q_{in}(t) = k_v\sqrt{h(t)} + \pi R^2 \frac{dh(t)}{dt} \tag{1.25}$$

En réécrivant le modèle dans (1.25) sous la forme d'un modèle d'état général défini dans (1.7), nous obtenons le modèle mathématique non linéaire du réservoir de stockage cylindrique vertical :

$$\begin{aligned} &\frac{dh(t)}{dt} = \frac{q_{in}(t) - k_v\sqrt{h(t)}}{\pi R^2}, \\ &y(t) = h(t), \end{aligned} \tag{1.26}$$

Le modèle mathématique non linéaire du réservoir cylindrique horizontal en temps discret peut être dérivé du modèle dynamique (1.26).

1.3 Linéarisation auteur d'un point de fonctionnement

Dès qu'un système est non linéaire, il est possible de le transformer en un système linéaire autour d'un point de fonctionnement. Ce dernier peut être défini comme un état des variables entrée/sortie qui vérifie l'équation différentielle et autour duquel on va étudier l'influence de petites variations des entrées sur la sortie. Dans notre cas, le point de fonctionnement sera (q_{in}^s, k_v, h^s) Nous allons illustrer les méthodes de linéarisation autour d'un point de fonctionnement à l'aide de l'équation suivante :

$$q_{in}^s = k_v\sqrt{h^s} \tag{1.27}$$

Soit le débit d'entrée q_{in}^s en régime permanent connu. Puis à partir de (1.27), nous pouvons évaluer le niveau du liquide en régime permanent comme suit :

$$h^s = \left(\frac{q_{in}^s}{k_v}\right)^2 \tag{1.28}$$

Le modèle d'états linéaire est exprimé par :

$$\begin{aligned} \dot{x}(t) &= \tilde{A}x(t) + \tilde{B}u(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{1.29}$$

Le Jacobien basé sur la relation suivante :

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} x^s, u^s \tag{1.30}$$

Où l'indice x^s, u^s indique que toutes les entrées de la matrice sont calculées aux points stationnaires (point de fonctionnement). Cependant, dans le cas des réservoirs présentés, le système est mono-variable et les matrices $\tilde{A}$, $\tilde{B}$, C et D sont données par :

$$\begin{aligned} \tilde{A} &= \left.\frac{df(.)}{dh^s}\right|_{h^s} \\ \tilde{B} &= \left.\frac{df(.)}{dq_{in}^s}\right|_{q_{in}^s} \\ C &= \left.\frac{dg(.)}{dh^s}\right|_{h^s} \\ D &= \left.\frac{df(.)}{dq_{in}^s}\right|_{q_{in}^s} \end{aligned} \tag{1.31}$$

Alors que le modèle mathématique linéaire en temps continu (1.29) est défini comme un ensemble d'équation différentielle du premier ordre et d'équation de sortie, la formulation en temps discret peut être exprimée sous la forme d'un ensemble d'équations aux différences et d'équation de sortie sous la forme suivante :

$$\begin{aligned} x(t+T_s) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{1.32}$$

Où les matrices A et B sont calculées en utilisant la propriété suivante :

$$e^{\begin{bmatrix} \tilde{A} & \tilde{B} \\ 0 & 0 \end{bmatrix} T_s} = \begin{bmatrix} M_{11} & M_{12} \\ 0 & I \end{bmatrix} \tag{1.33}$$

Alors :

$$\begin{aligned} A &= M_{11} \\ B &= M_{12} \end{aligned} \tag{1.34}$$

Les matrices C et D en temps discret sont les mêmes que dans (1.29b). Les matrices du système de temps discret (A, B) peuvent également être calculées via une discrétisation de maintien d'ordre zéro dans MATLAB.

Exemple : Linéarisation du réservoir de stockage cylindrique horizontal

Avec la condition initiale $h(0) = h^s$.

Le niveau du liquide dans le réservoir en régime permanent est connu. Ensuite, sur la base de l'équation (1.27), nous pouvons calculer le débit d'entrée en régime permanent. Le modèle mathématique linéarisé du réservoir de stockage cylindrique vertical défini par (1.29), est donné par :

$$\begin{aligned} \tilde{A} &= \frac{2q_{in}^s(h^s - R) - k_v h^{s3/2}}{4\ell(2Rh^s - h^{s2})^{3/2}}, \\ \tilde{B} &= \frac{1}{2\ell\sqrt{2Rh^s - h^{s2}}}, \\ C &= 1, \\ D &= 0. \end{aligned} \tag{1.35}$$

1.4 Résultats de simulation

La réponse indiciel d'un système représente le comportement temporel des sorties d'un système lorsque ses entrées changent d'une valeur à l'autre en peu de temps. Afin de simuler les réponses indicielles du modèle mathématique non linéaire en temps continu défini en (1.7), il est nécessaire de résoudre une équation différentielle pour la valeur particulière du signal d'entrée. Si l'on veut simuler les réponses indicielles d'un modèle mathématique linéaire en un temps continu, il est nécessaire de calculer les matrices $\tilde{A}$, $\tilde{B}$, C et D définies dans (1.31).

Dans cette section, nous analyserons les réponses indicielles du modèle mathématique non linéaire en temps continu pour les réservoirs conique, sphérique, stockage cylindrique horizontal et stockage cylindrique vertical [2].

Le signal de commande appliquée est le même pour tous les réservoirs (Figure 1.5).

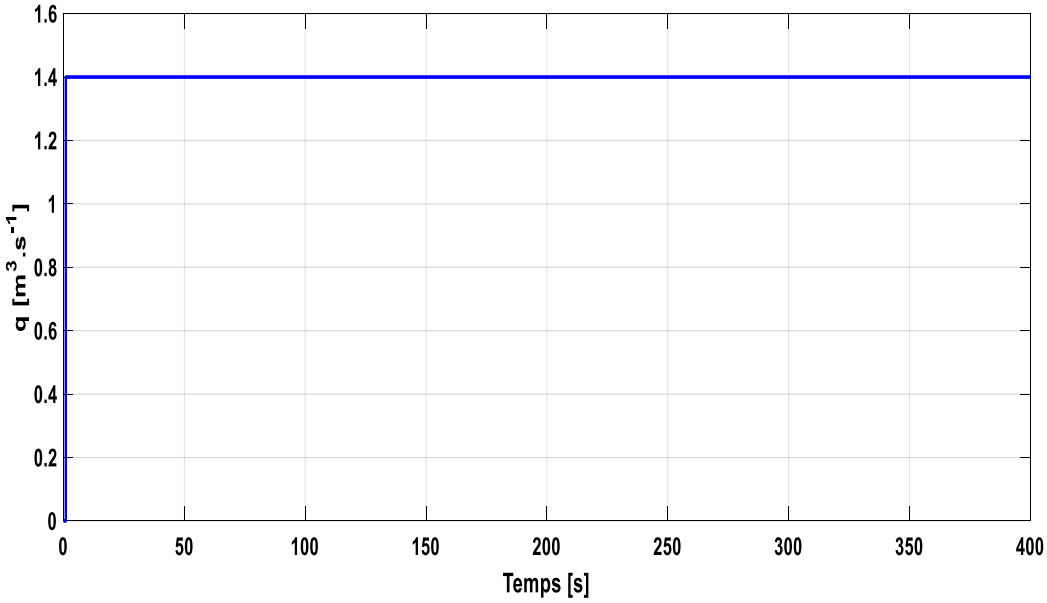


Figure 1.5 : La commande pour tous les réservoirs

1.4.1 Réservoir conique

Nous considérons un tronc inversé d'un cône droit comme un processus de réservoir conique caractérisé par des paramètres technologiques du Tableau 1.1.

Variable	Unité	Valeur
R_1	m	1.000
R_2	m	0.200
$h_{\max}$	m	2.000
k_v	$m^{2.5}S^{-1}$	0.75

Tableau 1.1 : Paramètres technologiques du réservoir de stockage conique

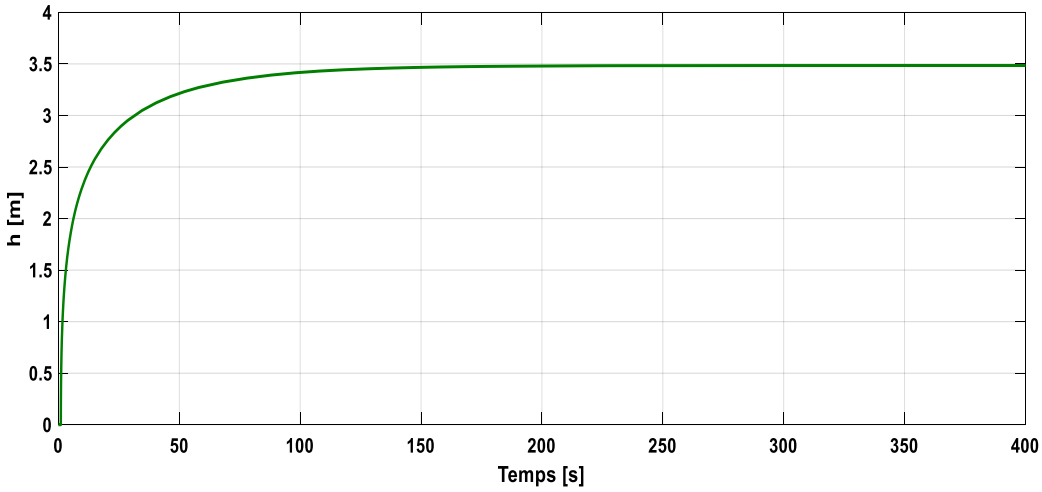


Figure 1.6 : Réponses du réservoir de stockage conique

La Figure 1.6 montre la réponse du réservoir conique, on remarque que le comportement dynamique du système est stable et ressemble au comportement d'un système du premier ordre.

1.4.2 Réservoir sphérique

Le processus considéré dans cette section est représenté par un réservoir de stockage sphérique avec les paramètres technologiques indiqués dans le Tableau 1.2.

variable	unité	Valeur
R	m	2.00
k_v	$m^{2.5}s^{-1}$	0.75

Tableau 1.2 Paramètres technologiques du réservoir de stockage sphérique

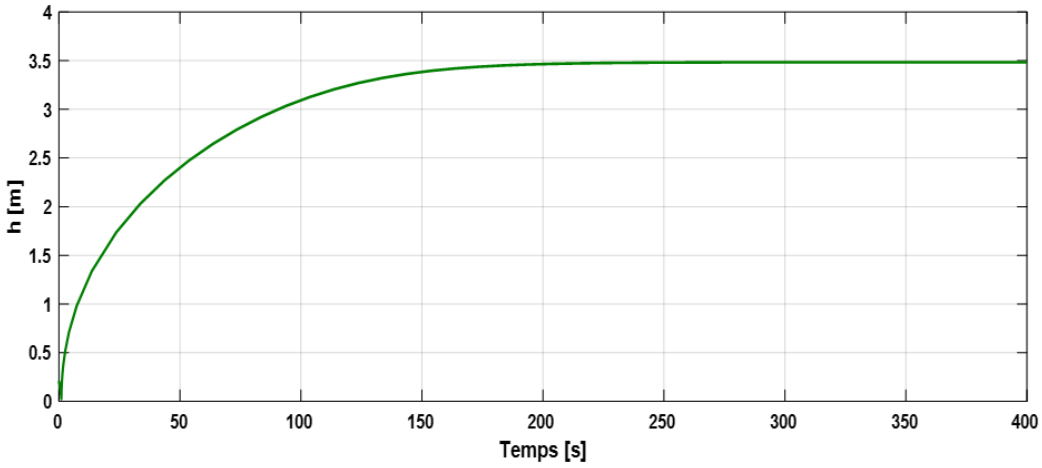


Figure 1.7 : Réponses indiciel du réservoir de stockage sphérique.

La Figure 1.7 montre la réponse du réservoir sphérique, on remarque que le comportement dynamique du système et moins rapide que celle du réservoir conique.

1.4.3 Réservoir cylindrique horizontal

Dans cette section, nous nous concentrerons sur le réservoir de stockage cylindrique horizontal qui est donné par les paramètres technologiques du Tableau 1.3.

Variable	Unité	Valeur
R	m	2.00
ℓ	m	4.00
k_v	$m^{2.5}s^{-1}$	0.75

Tableau 1.1 Paramètres technologiques du réservoir de stockage cylindrique horizontal.

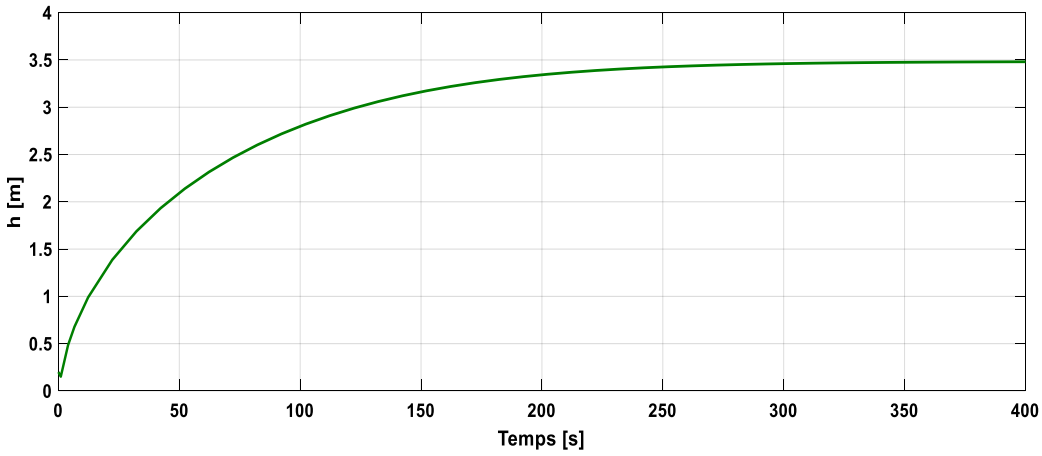


Figure 1.8 : Réponses indiciel du réservoir de stockage cylindrique horizontal.

La Figure 1.8 montre la réponse du réservoir de stockage cylindrique horizontal. On remarque que le comportement dynamique du système est presque similaire au réservoir sphérique, avec un temps de réponse plus grand.

1.4.4 Réservoir cylindrique vertical

Les paramètres technologiques du réservoir de stockage cylindrique vertical sont les même pour le réservoir de stockage cylindrique horizontal.

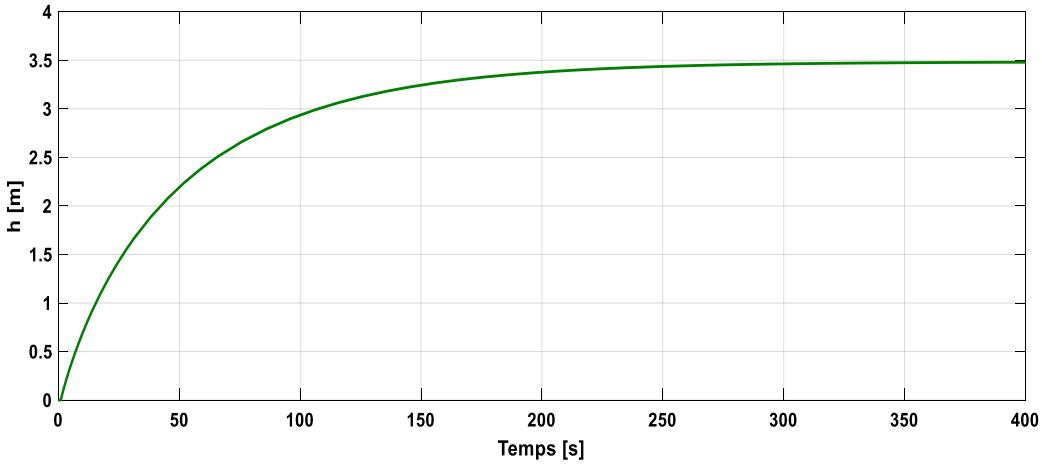


Figure 1.9 : Réponses indiciel du réservoir de stockage cylindrique vertical.

La Figure 1.9 montre la réponse du réservoir de stockage cylindrique vertical. On remarque que le comportement dynamique du système est rapide par rapport au réservoir cylindrique horizontal, avec un temps de réponse similaire.

1.5 Conclusion

Dans ce chapitre, nous avons présenté le modèle mathématique dynamique non linéaire des réservoirs conique, sphérique, cylindrique vertical et cylindrique horizontal en temps continue et discret. A la fin, nous avons donné un exemple sur la linéarisation d'un réservoir, et on a simulé les réponses indicielles des modèles mathématiques non linéaires en temps continu des différents réservoirs.

CHAPITRE 2

Généralité sur la régulation

2.1 Introduction

Le développement des outils mathématique, informatique et technologiques ont permet des nouvelles techniques avancées de commande, d'analyse et de supervision des processus industriels. Le contrôle des processus doit assurer un fonctionnement optimal sous des contraintes pratiques prédéfinis par les concepteurs selon un cahier de charge spécifique aux processus tel que la stabilité du système, la rapidité et la précision de la réponse dynamique.

Dans ce chapitre, nous allons présenter les notions de base de la régulation automatique et les diverses formes des régulateur PID.

2.2 Régulation automatique

La régulation automatique est l'ensemble des techniques qui permettent de contrôler une grandeur physique (température, pression, niveau, débit, pH, concentration, …), sans intervention humaine, pour la maintenir à une valeur donnée, appelée consigne. Cette grandeur physique est appelé grandeur réglée. La régulation est une rétroaction négative qui est utilisée dans un système asservi [3].

2.3 But de la régulation

La régulation ou l'asservissement d'un procédé a pour objectif le maintien le plus près possible de son optimum de fonctionnement, prédéfini par un cahier des charges (conditions ou performances imposées). Les aspects de sécurité du personnel et des installations sont à prendre en compte comme ceux concernant l'énergie et le respect de l'environnement. Le cahier des charges définit des critères qualitatifs à imposer qui sont traduits le plus souvent par des critères quantitatifs, comme par exemple, la stabilité, la précision et la rapidité [4].

2.4 Notion de boucle ouverte/fermée

2.4.1 Boucle ouverte

Dans la régulation en boucle ouverte, la variable de sortie ne provoque aucune influence sur la variable d'entrée (Figure 2.1).

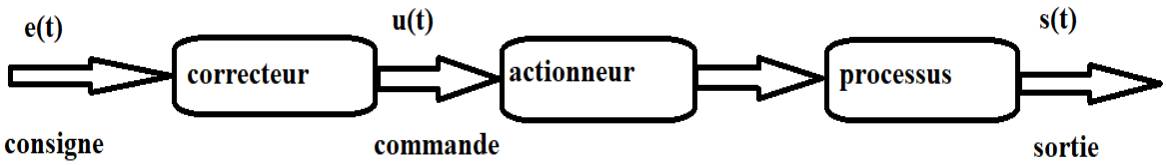


Figure 2.1 : Système en boucle ouverte

2.4.2 Boucle fermée

La régulation en boucle fermée représente un processus de commande dont la valeur mesurée est souvent surveillée et comparée par rapport à la consigne (avec feedback). Ainsi, elle provoque la génération d'un signal de commande afin d'ajuster la variable de commande à la consigne en dépit des perturbations.

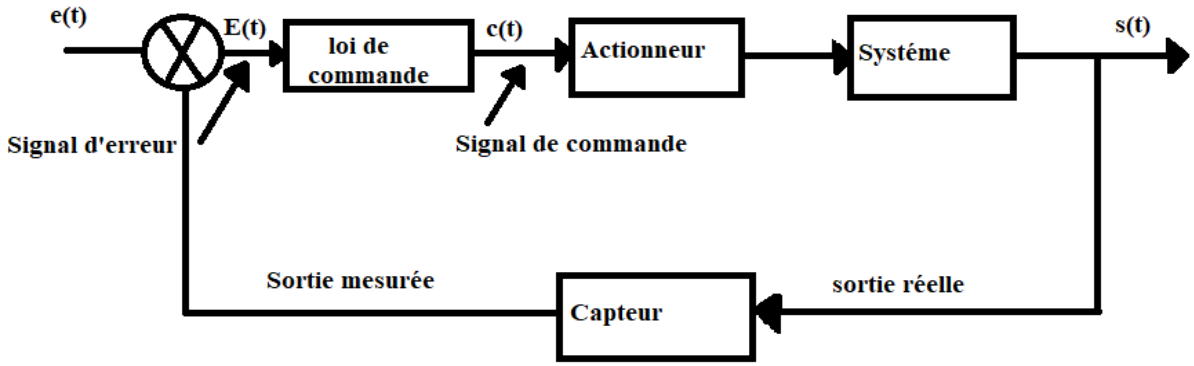


Figure 2.2 : Système en boucle fermée

2.5 Performances d'un système

Les performances d'un système asservi sont généralement caractérisées par les quatre critères suivants :

2.5.1 Stabilité

Cette condition est impérative mais avec un certain degré de stabilité (marge de sécurité). En général on impose une marge de gain de 2 à 2,5.

2.5.2 Précision

Le système doit posséder une bonne précision en régime permanent d'où une nécessite de mettre un régulateur PI, d'afficher un gain important dans le cas d'un régulateur P.

2.5.3 Rapidité

On demande en pratique que le système soit capable rapidement de compenser les perturbations et de bien suivre la consigne.

2.5.4 Dépassement

En général, on recommande un système de régulation dont le régime transitoire soit bien amorti et dont le dépassement ne dépasse pas 5% à 10% de la valeur nominal.

Un système asservi idéal est stable, précis et rapide. Cependant, il faut faire un compromis entre ces critères pour un système en boucle fermée.

2.6 Type de régulation automatique

2.6.1 Régulation tout ou rien (TOR)

Ce mode d'action est essentiellement discontinu. Sa réalisation impose de se fixer une limite inférieure et une limite supérieure pour le signal de commande.

2.6.2 Régulation analogique

C'est le mode ou le signal du régulateur et la mesure variant d'une manière continue dans le temps. Le mode d'action analogique le plus simple est l'action proportionnelle qui est réalisée par un régulateur (P). Généralement il est adéquat aux installations ayant une grande inertie.

2.6.3 Régulation numérique

Le principe de la régulation numérique est que le régulateur est programmé sur un microprocesseur et exécuté en temps réel, i. e. impérativement à chaque période d'échantillonnage.

2.7 Régulateur PID

Le régulateur PID est considéré comme un régulateur standard car c'est le plus utilisé dans l'industrie (proportionnel intégral dérivé) (Figure 2.3), permettant d'assurer à l'aide de ses trois paramètres les performances souhaitées (amortissement, temps de réponse, …) d'un processus modélisé par un deuxième ordre. Nombreux sont les systèmes physiques qui, même en étant complexes, ont un comportement voisin de celui d'un deuxième ordre. Par conséquent, le régulateur PID est bien adapté à la plupart des processus de type industriel et est relativement robuste par rapport aux variations des paramètres du procédé.

Si la dynamique dominante du système est supérieure à un deuxième ordre, ou si le système contient un retard important ou plusieurs modes oscillants, le régulateur PID n'est plus adéquat et un régulateur plus complexe (avec plus de paramètres) doit être utilisé, au dépend de la sensibilité aux variations des paramètres du procédé [3].

Le régulateur PID standard génère un signal de commande conformément à l'équation :

$$u(t)=K_p e(t)+\frac{K_p}{T_i}\int_0^t e(\tau)d\tau+K_p T_d \frac{de(t)}{dt} \qquad (2.1)$$

Avec $K_i=\frac{K_p}{T_i}$ et $K_d=K_p T_d$

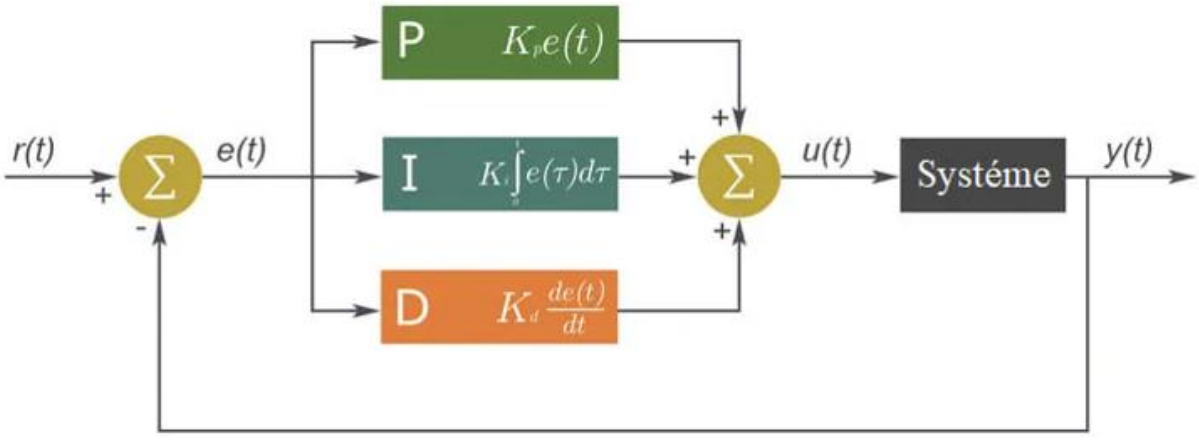


Figure 2.3 : Schéma bloc d'un régulateur PID

2.8 Formes des correcteurs PID

Plusieurs variantes de formulation du correcteur PID standard sont souvent employées dans le but d'augmenter ses performances ou d'en faciliter le dimensionnement.

2.8.1 Forme standard (forme mixte)

La forme standard de la fonction de transfert d'un correcteur PID est :

$$C(s)=\frac{u(s)}{e(s)}=K_p\left(1+\frac{1}{T_i s}+T_d s\right) \tag{2.2}$$

Le correcteur PID standard génère un signal de commande conformément à l'équation :

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(\tau)d\tau + K_p T_d \frac{de(t)}{dt} \tag{2.3}$$

2.8.2 Forme parallèle

La forme parallèle de la fonction de transfert d'un correcteur PID est :

$$C(s)=\frac{u(s)}{e(s)}=K_p+\frac{K_i}{s}+K_d s \tag{2.4}$$

Le correcteur PID parallèle génère un signal de commande conformément à l'équation :

$$u(t) = K_p e(t) + K_i\int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{2.5}$$

2.8.3 Forme série

La forme série de la fonction de transfert d'un correcteur PID est :

$$C(s)=\frac{u(s)}{e(s)}=K_p\left(1+\frac{1}{T_i s}\right)\left(1+T_d s\right) \tag{2.6}$$

Le correcteur PID série génère un signal de commande conformément à l'équation :

$$u(t) = K_p(1+\frac{T_d}{T_i})e(t) + \frac{K_p}{T_i}\int_0^t e(\tau)d\tau + K_p T_d \frac{de(t)}{dt} \tag{2.7}$$

2.9 Effet des actions du Régulateur PID

Sous forme d'un tableau récapitulatif (Tableau 2.1 et Tableau 2.2), on résume les points forts et les points faibles ainsi que les limitations des actions de base des régulateurs PID.

Action	Points forts	Points faibles
P	Action instantanée	Ne permet pas d'annuler une erreur statique mais permet de la réduire
I	Annule l'erreur statique	Action lente ralentit le système (effet déstabilisant)
D	Action très dynamique	Améliore la rapidité Apporte un effet stabilisant sensibilité aux bruits forte sollicitation de l'organe de commande

Tableau 2.1 : effet des action du correcteur PID

Augmentation de	Stabilité	Précision	Rapidité
K_p	diminue	augmente	augmente
T_i	augmente	Pas d'influence	diminue
T_d	diminue	Pas d'influence	augment

Tableau 2.2 : Tableau d'influence

2.10 Ajustement des paramètres PID

Nous allons présenter comment trouver les valeurs à attribuer aux trois coefficients (K_p, K_i, K_d) du régulateur PID. Il existe deux façons de procédé, la première par la modélisation, la deuxième par l'expérimentation. L'approche par l'expérimentation signifie que l'on va utiliser une réponse réelle du système pour régler d'abord grossièrement puis finement les coefficients du régulateur PID [5].

2.10.1 Approche par la modélisation

Le choix de procéder à la modélisation du système ou non est dicté par les contraintes inhérentes au système. Souvent, la complexité des systèmes réels place la modélisation hors d'atteinte, mais dans certain cas, en chimie ou en mécanique par exemple, les règles qui régissent le système sont suffisamment simples est particulièrement critique ou difficile d'accès (processus industriel lourd, système qui ne peut être mis hors service). Il est alors indispensable de modéliser le système afin d'avoir un jeu de coefficients suffisamment précis pour obtenir d'emblée un régulateur PID qui soit fonctionnel.

2.10.2 Approche expérimentale

Ziegler et Nichols ont proposé deux approches expérimentales destinées à fixer rapidement les paramètres des correcteurs P, PI et PID. La première nécessite l'enregistrement de la réponse indicielle du système en boucle ouverte, alors que la deuxième exige d'amener le système en boucle fermée à sa limite de stabilité.

2.10.2.1 Approche de Ziegler-Nichols pour les systèmes en boucle ouverte

Cette méthode est conforme aux systèmes dont la réponse indicielle en boucle ouverte peut être relevée et ne présente pas de dépassement (réponse apériodique). Les correcteurs proposés ont été calculés pour convenir pour un système à commander de fonction de transfert $H(s)=\frac{Ke^{-T_u s}}{1+T_a s}$, les paramètres K, T_u et T_a sont relevés sur la réponse du système réel à commander a un échelon d'amplitude A selon la Figure 2.3. Notons que d'autres méthodes d'identification peuvent être utilisées pour le calcul des valeurs des paramètres du modèle du 1er ordre.

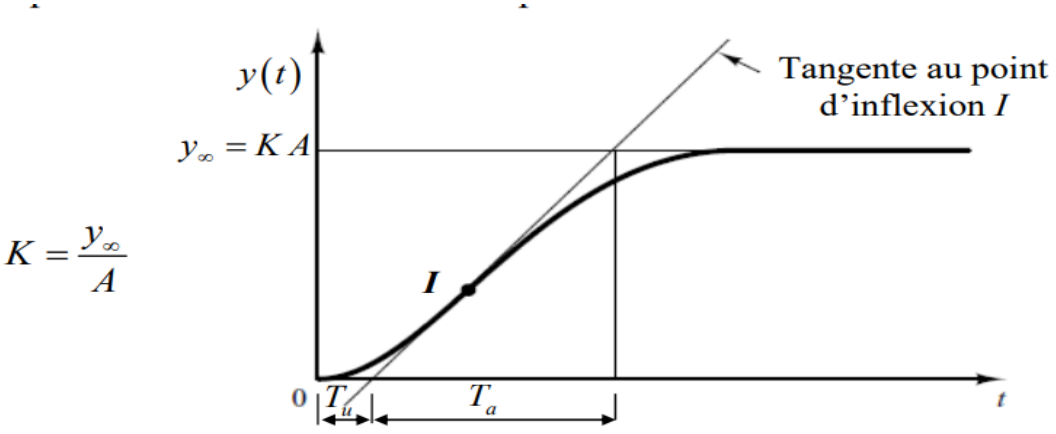


Figure 2.4 : Réponse indicielle en boucle ouverte

Ziegler et Nichols proposent de calculer les paramètres des correcteurs P, PI ou PID à l'aide des recommandations suivantes

Paramètre du correcteur	K_p	T_i	T_d
P C(s)= K_p	$\frac{1}{K}\frac{T_a}{T_u}$	-	-
PI C(s)= $K_p\left(1+\frac{1}{T_i s}\right)$	$\frac{0.9}{K}\frac{T_a}{T_u}$	$3.3T_u$	-
PID C(s)= $K_p\left(1+\frac{1}{T_i s}\right)\left(1+T_d s\right)$	$\frac{1.2}{K}\frac{T_a}{T_u}$	$2T_u$	$0.5T_u$

Tableau 1.3 : Méthode de Ziegler-Nichols en boucle ouverte

Procédure :

- Obtenir la réponse indicielle en boucle ouverte du système a commandé
- Déterminer le temps T_u , T_a et K à partir de cette réponse indicielle
- Calculer les paramètres du correcteur choisi à partir du tableau
- Si nécessaire, ajuster les paramètres du correcteur

Remarque : en général, les gain proportionnels K_p préconisés par Ziegler-Nichols sont trop importants et conduisent à un dépassement supérieur à 20 %. Il ne faut pas craindre de réduire ces gains d'un facteur 2 pour obtenir une réponse satisfaisante.

2.10.2.2 Approche de Ziegler-Nichols en boucle fermée

L'essai en boucle ouverte est irréalisable pour certains systèmes pour deux raison, parce qu'ils sont instables, ou parce que le gain en boucle ouverte est grand au point de provoquer des variations de la sortie dangereuses pour l'installation. On peut alors utiliser une méthode basée sur un essai en boucle fermée, dite méthode du pompage limite (oscillations entretenues). La deuxième méthode de Ziegler-Nichols repose sur la connaissance des paramètres du point critique (K_{cr} et T_{cr}).

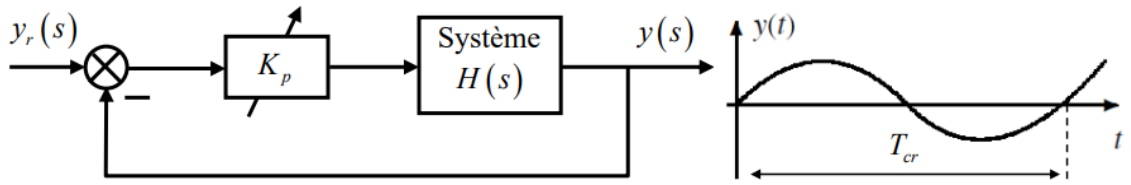


Figure 2.5 : Essai d'instabilité en boucle fermée

Ziegler et Nichols proposent de calculer les paramètres des correcteur P, PI ou PID à l'aide des recommandations suivantes :

Paramètres correcteur	Kp	Ti	Td
P	0.5 K_{cr}	-	-
PI	0.45 K_{cr}	0.83 T_{cr}	-
PID	0.6 K_{cr}	0.5 T_{cr}	0.125 T_{cr}

Tableau 2.4 : Méthode de Ziegler-Nichols en boucle fermée

Procédure :

- Placer le système en boucle fermée avec un correcteur proportionnel de gain K_p
- Augmenter progressivement le gain K_p jusqu'à obtenir une réponse périodique
- Relever la valeur K_{cr} du gain K_p ainsi atteinte, et la période T_{cr} des oscillations de la sortie
- calculer les paramètres des correcteurs choisis à partir du tableau
- si nécessaire, ajuster les paramètres du correcteur

Remarque : on note que les deux méthodes de Ziegler-Nichols conduisent à des valeurs très proches pour les paramètres du correcteur et par conséquent les performances seront similaires.

2.10.3 Méthode des approximations successives

Elle s'applique sur les systèmes apériodiques en boucle fermée. Cette méthode consiste de réglée successivement l'action proportionnel, l'action intégrale et enfin l'action dérivée.

Le mode opératoire est décrit par les étapes suivantes :

- On annule l'action intégrale et dérivée en mettant $T_i = \infty$, $T_d = 0$.
- On augmente K_p jusqu'à avoir la réponse la plus rapide avec un amortissement maximum et un écart minimum (voir Figure 2.6).
- On diminue T_i pour annuler l'erreur statique et avoir la réponse la plus rapide avec un amortissement maximum et un écart minimum (voir Figure 2.7).
- On augmente T_d jusqu'à avoir la réponse optimal (voir Figure 2.8).

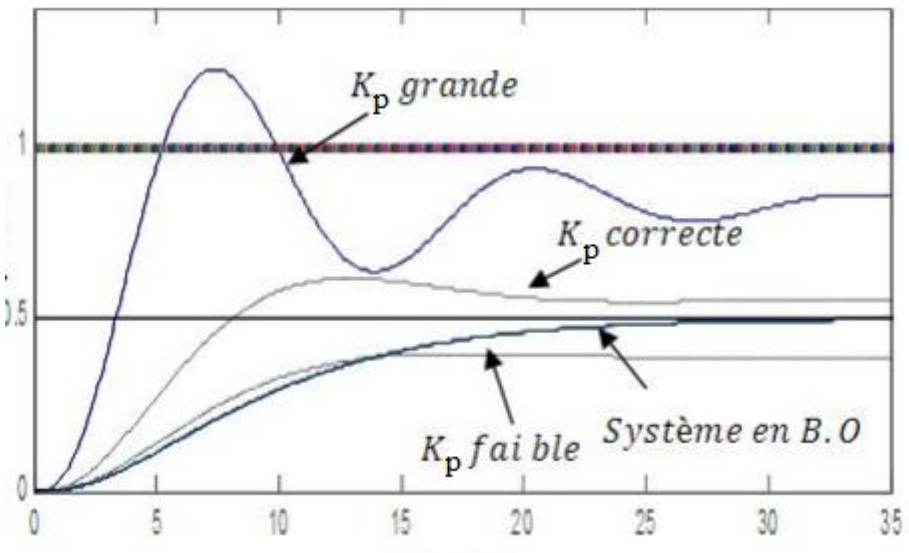


Figure 2.6 : Détermination du gain proportionnel par la méthode d'approximations successives.

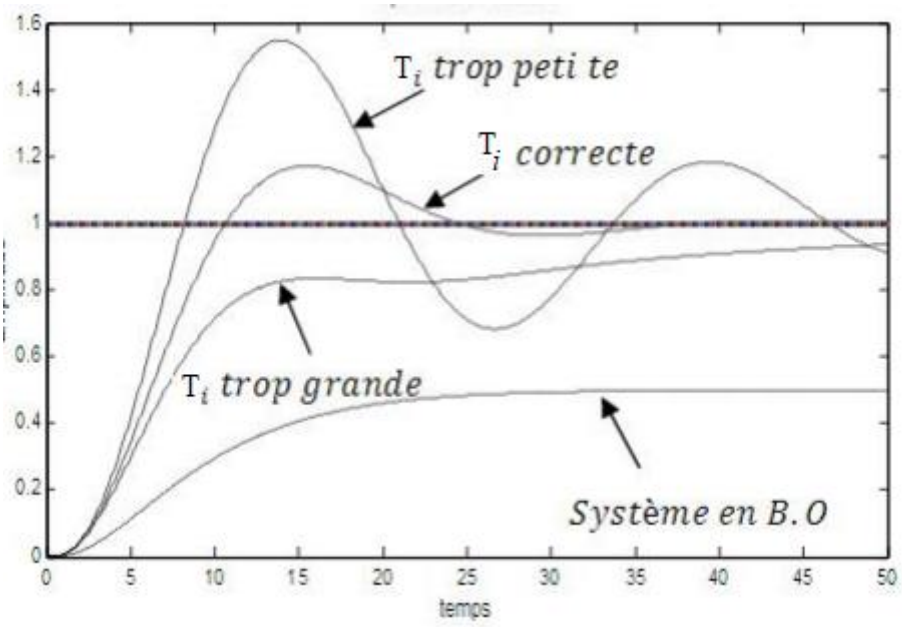


Figure 2.7 : Détermination de la constante de temps intégrale par la méthode d'approximations successives.

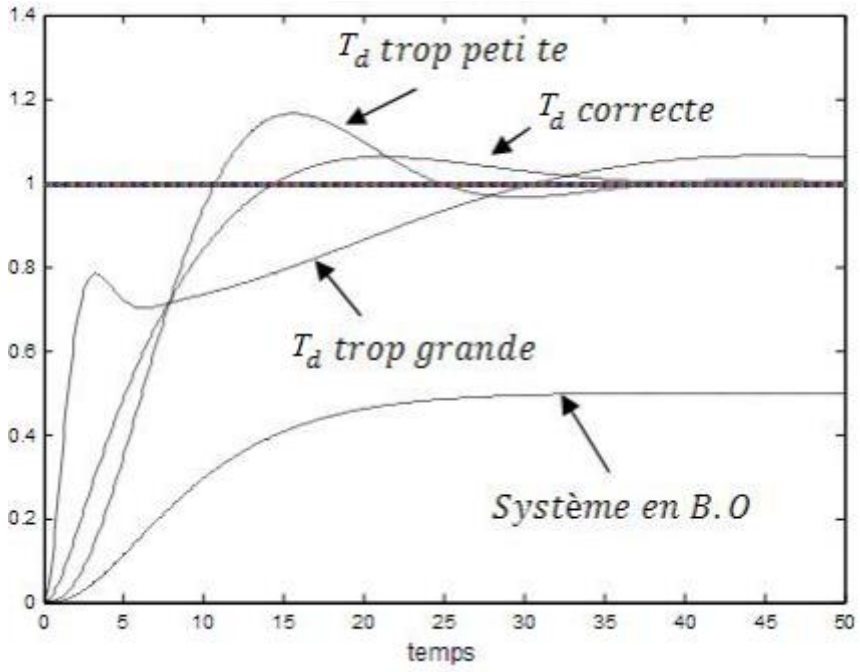


Figure 2.8 : Détermination de la constante de temps dérivée par la méthode d'approximations successives.

Cette méthode est très utilisée vue sa simplicité. Son principal avantage est qu'elle ne nécessite pas la connaissance du modèle du processus. L'inconvénient majeur est que son application devient longue sur les processus à grande inertie [6].

2.11 Discrétisation de régulateur PID

Pour une implémentation sur calculateur, il est nécessaire de faire la transposition de la fonction de transfert du PID continu C(s) en un correcteur numérique C(z) pour avoir un algorithme de PID numérique. On peut distinguer trois approximations pour faire la discrétisation du PID [7] :

- **Approximation avancée :**

On a : $$\dot{X}(t) = \frac{X(t+T_e) - X(t)}{T_e} \tag{2.8}$$

La transformée de Laplace : $$SX(S) = \frac{X(S)(e^{ST_e} - 1)}{T_e} \tag{2.9}$$

Donc : $$S = \frac{e^{ST_e} - 1}{T_e} \quad \text{avec} \quad Z = e^{ST_e}$$

On obtient : $$S = \frac{Z-1}{T_e} = \frac{1 - Z^{-1}}{T_e Z^{-1}} \tag{2.10}$$

Alors : $$C(Z) = K_p\left(1 + \frac{1}{T_i}\frac{T_e z}{Z-1} + \frac{N(Z-1)}{(1 + \frac{NT_e}{T_d})z - 1}\right) \tag{2.11}$$

- **Approximation retardée :**

On a : $$\dot{X}(t) = \frac{X(t) - X(t-T_e)}{T_e} \tag{2.12}$$

La transformée de Laplace : $$SX(S) = \frac{X(S)(e^{ST_e} - 1)}{T_e} \tag{2.13}$$

Donc : $$S = \frac{1 - e^{-ST_e}}{T_e} \tag{2.14}$$

$$S = \frac{Z-1}{T_e Z} = \frac{1-Z^{-1}}{T_e} \tag{2.15}$$

Alors : $$C(Z) = K_p(1 + \frac{1}{T_i}\frac{T_e}{Z-1} + \frac{N(Z-1)}{z-(1-\frac{NT_e}{T_d})}) \tag{2.16}$$

- **Approximation de TUSTIN :**

La méthode de la transformation bilinéaire correspond au développement suivant lorsque T_e (le temps de discrétisation) converge vers 0

$$Z = e^{ST_e} \Leftrightarrow S = \frac{1}{T_e}\ln(Z)$$

$$S = \frac{2}{T_e}[\frac{Z-1}{Z+1} + \frac{1}{3}(\frac{Z-1}{Z+1})^3 + \frac{1}{5}(\frac{Z-1}{Z+1})^5 .....] \tag{2.17}$$

L'approximation bilinéaire du premier ordre est :

$$S = \frac{2}{T_e}\frac{Z-1}{Z+1} = \frac{2}{T_e}\frac{1-Z^{-1}}{1+Z^{-1}} \tag{2.18}$$

Alors : $$C(Z) = K_p(1 + \frac{1}{2T_i}\frac{Z+1}{Z-1} + \frac{N(Z-1)}{(1+\frac{NT_e}{2T_d})z-(1-\frac{NT_e}{2T_d})}) \tag{2.19}$$

Dans notre cas, nous avons choisi l'approximation de TUSTIN pour discrétisé l'équation du régulateur PID standard

$$C(Z^{-1}) = K_p + \frac{K_p T_e}{2T_i}\frac{1+Z^{-1}}{1-Z^{-1}} + \frac{2K_p T_d}{T_e}\frac{1-Z^{-1}}{1+Z^{-1}} \tag{2.20}$$

On obtient les trois commandes suivantes :

$$\begin{aligned} U_P(Z^{-1}) &= K_p.e(Z^{-1}) \\ U_I(Z^{-1}) &= K_i.\frac{1+Z^{-1}}{1-Z^{-1}}.e(Z^{-1}) \\ U_D(Z^{-1}) &= K_d.\frac{1-Z^{-1}}{1+Z^{-1}}.e(Z^{-1}) \end{aligned} \tag{2.21}$$

Avec :

$$K_i = \frac{K_p T_e}{2T_i}$$
$$K_d = \frac{2K_p T_d}{T_e} \quad (2.22)$$

Les équations a programmé sont :

$$U_P(K) = K_p.e(K)$$
$$U_I(K) = U_i(K-1) + K_I.e(K) + K_I.e(K-1) \quad (2.23)$$
$$U_D(K) = -U_d(K-1) + K_d.e(K) - K_d.e(K-1)$$

Avec U(k) C'est la sortie de notre bloc PID définie comme suit :

$$U(K) = U_P(K) + U_I(K) + U_D(K) \quad (2.24)$$

2.12 Conclusion

Dans ce chapitre, nous avons présenté des notions générales sur la régulation automatique et ces différentes méthodes soit en boucle ouverte ou en boucle fermée.

Aussi nous avons présenté le régulateur PID qui est le plus utilisés et ce pour plusieurs raisons. Premièrement, il est très simple à mettre en place et s'avère efficace pour la plupart des systèmes réels. De plus, le calcul des coefficients laisse le choix entre plusieurs méthodes de difficulté croissante.

CHAPITRE 3

Automates programmable industriel

3.1 Introduction

L'automate programmable industriel (API) est l'un des appareils de commande les plus utilisés dans le domaine industriel. Les API sont apparues pour la première fois à la fin des années soixante aux Etats-Unis dans le secteur de l'industrie automobile.

Dans ce chapitre, on regroupe toutes les informations sur les API avec les équipements essentiels et les technique utilisées, afin d'assurer un fonctionnement continuelle.

Ce chapitre présente, en particulier, une description des automates programmables industriels de Siemens et des logiciels utilisés pour la programmation de ces automates et pour la conception des interfaces homme-machine (Step 7 et WinCC flexible).

3.2 Système automatisé

Un système automatisé est un ensemble d'éléments en interaction, et organisés dans un but précis : agir sur une matière d'œuvre pour lui donner une valeur ajoutée (Figure 3.1). Le système automatisé est soumis à des contraintes : énergétiques, de configuration, déréglage et d'exploitation qui interviennent dans tous les modes de marche et d'arrêt du système [3].

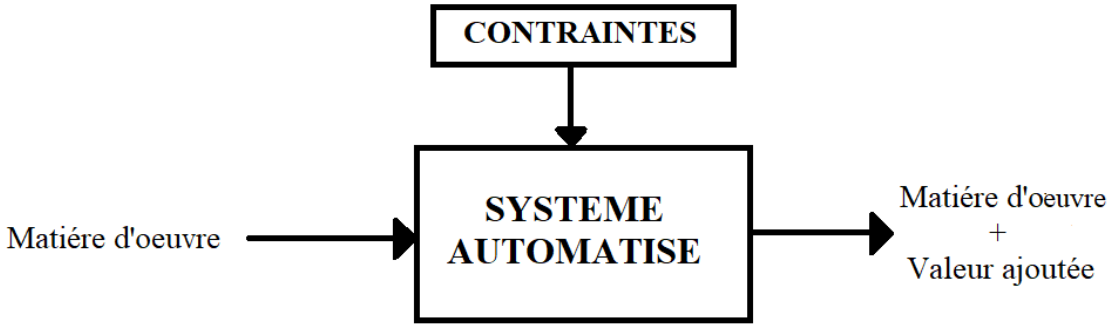


Figure 3.1 : Représentation d'un automate programmable industriel.

3.2.1 Objectif de l'automatisation

L'objectif de l'automatisation des systèmes est de produire, en ayant recours le moins possible à l'homme, des produits de qualité et ce pour un coût le plus faible possible, avec une sécurité assurée et une possibilité d'adaptation à des contextes particulier.

Elle permet de :

- Eliminer les tâches répétitives.
- Simplifier le travail de l'homme.
- Augmenter la sécurité (responsabilité).
- Accroître la productivité.

3.2.2 Avantages et inconvénients de l'automatisation

Les avantages

- S'adapte facilement à tous les milieux de production.
- La souplesse d'utilisation.
- La création des postes pour les automaticiens.
- Accélération des processus de production, en gardant un produit de qualité.

Les inconvénients

- La suppression d'emplois.
- Le coût élevé du matériel, principalement avec les systèmes hydrauliques.
- La maintenance doit être structurée.

3.2.3 Structure d'un système automatisé

Dès sa conception, un système à automatiser doit être décomposé en trois parties (Figure 3.2) :

La partie opérative :

C'est la partie visible du système. Elle comporte les éléments du procédé c'est-à-dire :

- des pré-actionneurs (distributeurs, contacteur) qui reçoivent des ordres de la partie commande.
- des actionneurs (vérins, moteurs, vannes) qui ont d'exécuter ces ordres, ils transforment l'énergie pneumatique, hydraulique, ou électrique en énergie mécanique.
- des capteurs qui informent la partie commande de l'exécution du travail.

La partie commande :

Cette partie de l'automatisme gère selon une suite logique le déroulement ordonné des opérations à réaliser. Il reçoit des informations en provenance des capteurs de la partie opérative, et les restitue vers cette même partie opérative en direction des pré-actionneurs et actionneurs.

La partie relation :

Sa complexité dépend de l'importance du système. Elle regroupe les différentes commande nécessaires au bon fonctionnement du procédé, c'est-à-dire marche/arrêt [10].

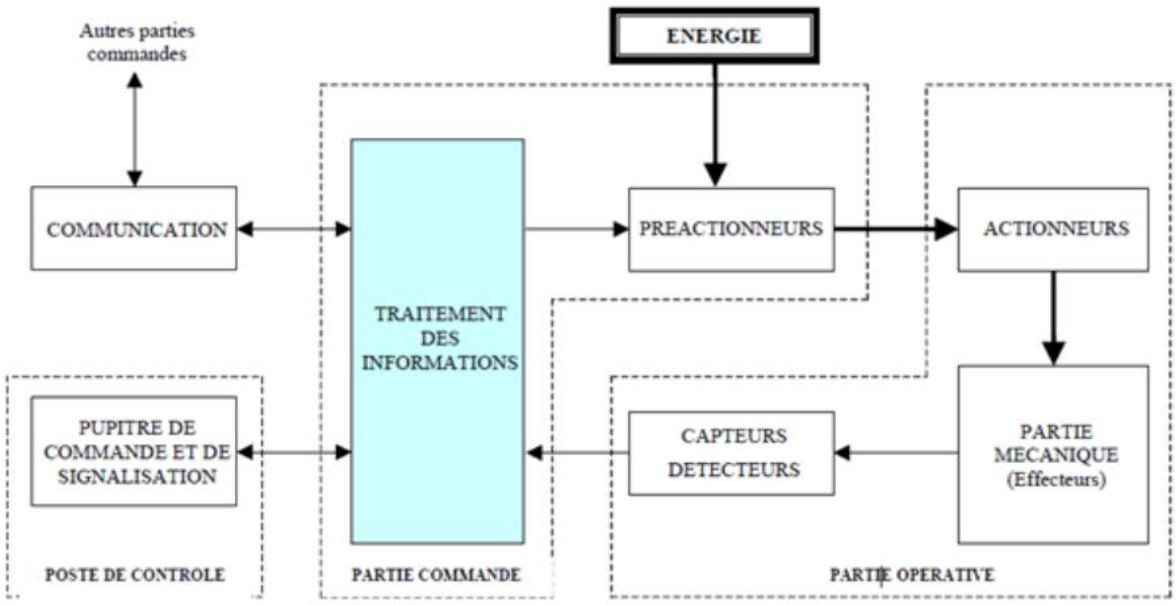


Figure 3.2 : Structure d'un système automatisé [12].

3.3 Automates programmables industriels

Un automate programmable industriel (API) est un appareil électronique spécialisée dans la conduite et la surveillance en temps réel des processus industriels.

Il exécute une suite d'instructions introduites dans sa mémoire sous forme de programme et s'apparente par conséquent aux machines de traitement de l'information.

Trois caractéristiques fondamentales le distinguent totalement des outils d'informatiques tels que les ordinateurs utilisés dans les entreprises :

- Connexion directe aux différents capteurs et actionneurs grâce à ces entrées/sortie.
- Fonctionnement dans des conditions industrielles sévères (température, vibrations, humidité, microcoupure de l'alimentation en énergie électrique…).
- Son aspect pratique grâce à la possibilité de sa programmation en utilisant un langage spécialement développé pour les automates (Step 7) [8].

3.3.1 Principe de fonctionnement d'un automate programmable

Tous les automates fonctionnent selon le même mode opératoire (Figure 3.3) :

- **Traitement interne :** l'automate effectue des opérations de contrôles et met à jour certains paramètres systèmes (détection des passages en RUN/STOP, mises à jour des valeurs de l'horodateur,…).
- **Lecture des entrées :** l'automate lit les entrées (de façon synchrone) et les recopies dans la mémoire image des entrées.
- **Exécution du programme :** l'automate exécute le programme instruction par instruction et écrit les sorties dans la mémoire image des sorties.
- **Ecriture des sorties :** l'automate bascule les différentes sorties (de façon synchrone) aux positions définies dans la mémoire image des sorties. Ces quatre opérations sont effectuées continuellement par l'automate (fonctionnement cyclique) **[**10].

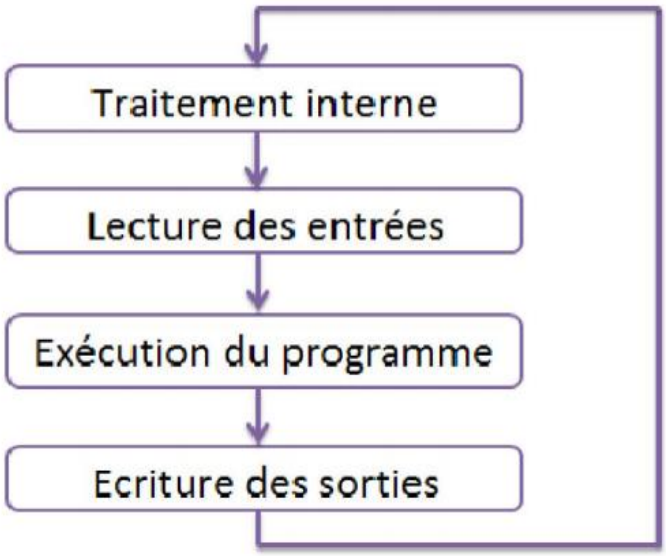


Figure 3.3 : Fonctionnement d'un API [8].

3.3.2 Architecture d'un API

3.3.2.1 Aspects externes

Les automates peuvent être de type compact ou modulaire.

Compacte :

Il intègre le processeur, l'alimentation et les entrées/sortie. Il peut réaliser certaines fonctions supplémentaires et recevoir des extensions limitées. Il est généralement destiné à la commande de petits automatismes.

Modulaire :

Dans ce modèle, le processeur, l'alimentation et les interfaces entrées/sorties résident dans des unités séparées (modules). Ces automates sont intégrés dans les automatismes complexes de grande puissance et de capacité de traitement.

3.3.2.2 Aspect interne

Un automate programmable industriel est donc constitué de :

Module d'alimentation :

Le module d'alimentation transforme l'énergie externe provenant du réseau en la mettant en forme afin de fournir aux différents modules de l'API les niveaux de tension nécessaires à leur bon fonctionnement. Plusieurs niveaux de tension peuvent être utilisés par les circuits internes (3V, 5V, 12V, 24V…). Il sera dimensionné en fonction des consommations des différentes parties.

Unité de traitement ou processeur :

Le processeur gère l'ensemble des échanges informationnels en assurant :

- La lecture des informations d'entrée.
- L'exécution des instructions du programme mis en mémoire.
- La commande ou l'écriture des sorties.

Mémoire programme :

La mémoire programme de type RAM contient les instructions à exécuter par le processeur afin de déterminer les ordres à envoyer aux pré-actionneurs reliés à l'interface de sortie en fonction des informations recueillies par les capteurs reliés à l'interface d'entrée.

Mémoire de données :

La mémoire de donnée permet le stockage de :

- L'image des entrées reliées à l'interface d'entrée.
- L'état des sorties élaborées par le processeur.
- Les valeurs internes utilisées par le programme (résultats de calculs, états intermédiaires…).
- Les états forcés ou non des entrées/sorties.

Interface d'entrée :

L'interface d'entrée permet la connexion à l'API d'un multiple de capteurs pouvant être de type

- TOR (logiques ou Tout Ou Rien).
- Numériques.
- Analogiques.

Ces différentes entrées sont mises en forme par l'interface d'entrée avant d'être stockées dans la mémoire de données.

Interface de sortie :

L'interface de sortie permet la connexion de l'API aux pré-actionneurs, qui peuvent être de type

- TOR (logiques ou Tout Ou Rien).
- Numériques.
- Analogiques. [9]

3.3.3 Critère de choix d'un API

Pour choisir un API, l'automaticien doit réunir certaines informations qui sont :

- Le nombre et la nature des entrées et des sorties.
- Le type de programme de fonctionnement désiré.
- La nature de traitement (temporisation, couplage, comptage …), qui va permettre de choisir l'unité centrale ainsi que la taille de la mémoire à utiliser.
- Le langage du dialogue (la console détermine le langage de programmation).
- Le type de communication avec d'autre système.
- La fiabilité et la robustesse.
- La vitesse du travail. [11]

3.4 Présentation de la gamme SIMATIC

3.4.1 SIMATIC S7

Dans la gamme S7 on distingue cinq grandes familles d'automates programmables industriels décrites dans ces paragraphes qui suivent :

3.4.1.1 SIMATIC S7-200

La famille S7-200 est constituée de micro-automates programmables utilisables dans des applications d'automatisations variées. La Figure 3.4 présente un micro-automate S7 -200. Son dessin compact, ses possibilités d'expansion, son faible prix et son important jeu d'opérations en font une solution idéale pour la commande de petites applications. On outre, le large choix de tailles et de tensions de CPU offre la souplesse nécessaire pour résoudre un problème d'automatisation.

Figure 3.4 : API SIEMENS S7-200

Un automate programmable S7-200 consiste en une CPU S7-200 seule ou complétée de divers modules d'extension facultatifs connectés à cette dernière à l'aide d'un connecteur de bus fourni avec ce module d'extension [8, 12].

3.4.1.2 SIMATIC S7-300

La famille S7-300 est constituée d'automates programmables de conception modulaire utilisés pour des automatismes de gamme moyenne, et peuvent être connectés entre eux au moyen d'un câble-bus PROFIBUS.

Un automate S7-300 consiste en une CPU, un module d'alimentation PS, un module de comptage FM, un module de signaux SM et un processeur de communication. Comme indiqué dans la Figure 3.5 [8, 12].

Figure 3.5: API SIEMENS S7-300

3.4.1.3 SIMATIC S7-400

La famille S7-400 est aussi constituée d'automates programmables de conception modulaire. Pratiquement chaque tâche d'automatisation peut être résolue par un choix approprié des constituants de S7-400 et avec la possibilité d'expansion de plusieurs modules.

Les modules se présentent sous forme de boitiers que l'on adapte sur un châssis [8, 12].

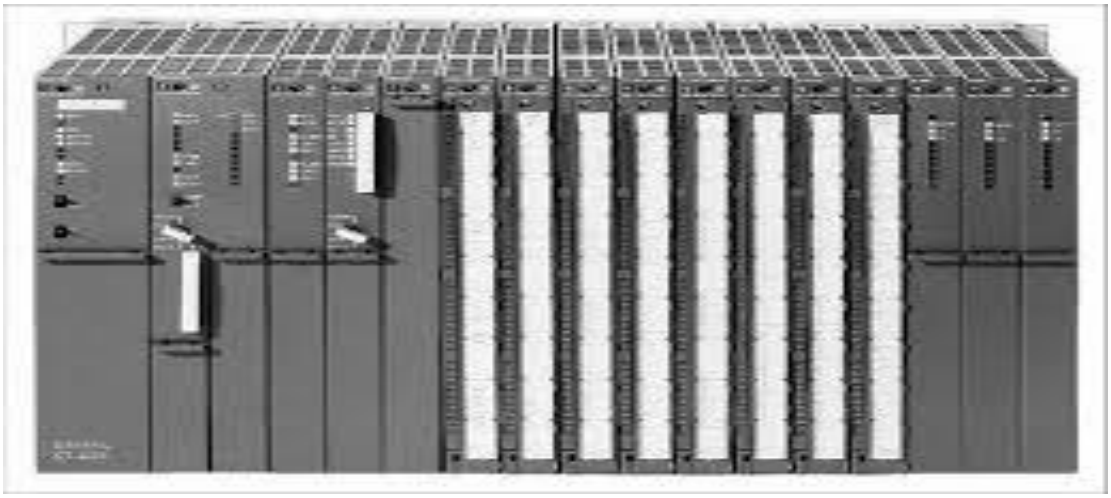


Figure 3.6: API SIEMENS S7-400

3.4.1.4 SIMATIC S7-1200

Le contrôleur S7-1200 offre la souplesse et la puissance nécessaires pour commander une large gamme d'appareils afin de répondre à vos besoins en matière d'automatisation. Sa forme compacte, sa configuration souple et son important jeu d'instructions en font une solution idéale pour la commande d'applications très variées.

La CPU combine un microprocesseur, une alimentation intégrée, des circuits d'entrée et de sortie, un PROFINET intégré, des E/S rapides de commande de mouvement, ainsi que des entrées analogiques intégrées dans un boîtier compact en vue de créer un contrôleur puissant. Une fois que vous avez chargé votre programme, la CPU contient la logique nécessaire au contrôle et à la commande des appareils dans votre application. La CPU surveille les entrées et modifie-les sorties conformément à la logique de votre programme utilisateur, qui peut contenir des instructions booléennes, des instructions de comptage, des instructions de temporisation, des instructions mathématiques complexes ainsi que des commandes pour communiquer avec d'autres appareils intelligents [13].

Figure 3.7: API SIEMENS S7-1200

3.4.1.5 SIMATIC S7-1500

Le système d'automatisation SIMATIC S7-1500 offre la flexibilité et la puissance nécessaires à un large éventail d'applications d'automatisation dans la construction de machines et d'installations. Sa structure modulable vous permet d'adapter votre automate aux conditions sur site.

Les CPU technologiques SIMATIC S7-1500 proposent, outre les fonctions technologies et Motion Control présentent par défaut sur les S7-1500, des fonctionnalités supplémentaires, comme les fonctions élargies de synchronisme et de profils de cames.

Le système d'automatisation SIMATIC S7-1500 est conforme à l'indice de protection IP20 et conçu pour être installé dans une armoire électrique dans un environnement sec.

Les CPU SIMATIC S7-1500R/H (CPU redondantes ou à haute disponibilité) vous permettent d'accroître la disponibilité de votre installation. Le programme utilisateur est traité de manière synchrone sur 2 CPU pour pouvoir passer de la CPU principale à la CPU réserve en cas de besoin [14].

Figure 3.8 : API SIEMENS S7-1500

3.4.2 Description de l'automate S7-300

Le S7-300 est un automate modulaire d'une gamme excellente des produits SIEMENS, il est synonyme de la nouvelle technologie des automates programmables. Le S7-300 est utilisé dans presque toutes les branches de l'industrie où les applications les plus variées. Sa modularité permet de réaliser des fonctions d'automatisations les plus diverses à partir des différents modules. Ses principales caractéristiques son :

- Sa puissance et sa rapidité.
- La possibilité d'intégration de nouvelles taches.
- Haute performance grâce aux nombreuses fonctions intégrées [8].

3.5 Présentation du logiciel de programmation STEP 7

STEP 7 fait partie de l'industrie logicielle SIMATIC (Figure 3.9). Le logiciel STEP 7 permet de concevoir, configurer, programmer, tester, mettre en service et maintenir les systèmes d'automatisation SIMATIC à base de S7-300 et S7-400. Les tâches de bases qu'il offre à son utilisateur lors de la création d'une solution d'automatisation sont :

- La création et gestion de projets.
- La configuration et le paramétrage du matériel et de la communication.
- La gestion des mnémoniques.
- La création des programmes.
- Le chargement de programmes dans les systèmes cibles.

- Le test de l'installation d'automatisation.
- Le diagnostic lors des perturbations dans l'installation. [8]

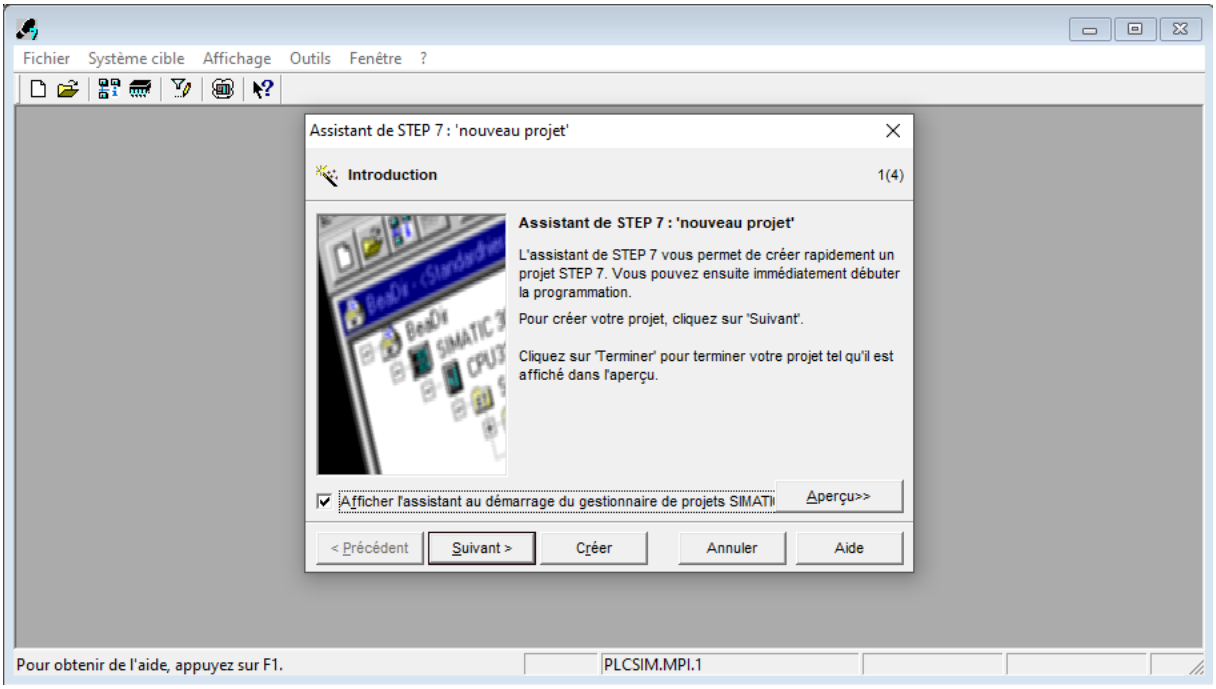


Figure 3.9 : Interface de Logiciel STEP7

3.5.1 Langage de programmation

Il existe plusieurs types de langage de programmation.

3.5.1.1 Langage LIST (Liste d'instruction)

Le langage de programmation LIST (liste d'instructions) est un langage textuel proche du langage machine. Chaque instruction correspond à une étape de l'exécution du programme par la CPU. Vous pouvez regrouper plusieurs instructions en réseaux.

Le langage de programmation LIST fait partie du logiciel de base STEP 7. Il vous permet d'éditer des blocs S7 avec des éditeurs incrémentaux ou de créer votre programme dans une source LIST avec un éditeur orienté source, puis de le compiler en blocs.

Exemple :

```
A(
O     I     1.5
O
AN    I     6.3
AN    I     1.3
)
AN    I     2.2
=     Q     3.3
```

Figure 3.10 : Exemple de programme List

3.5.1.2 Langage CONT (contacte ou Ladder)

Dans le langage de programmation graphique CONT, la représentation est fondée sur des schémas à relais. Les éléments d'un tel schéma, comme par exemple les contacts à ouverture ou les contacts à fermeture sont reliés pour former des réseaux. Un ou plusieurs de ces réseaux forment la section d'instructions complète d'un bloc de code.

Le langage de programmation CONT fait partie du logiciel de base STEP 7.

Exemple :

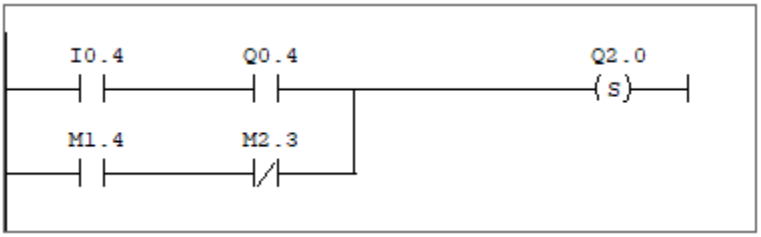


Figure 3.11 : Exemple de programme Ladder

Dans le langage CONT, vous créez le programme en utilisant un éditeur incrémental.

3.5.1.3 Langage LOG (logigramme)

Le langage de programmation LOG utilise les pavés logiques bien connus dans l'algèbre booléenne pour la représentation logique. Il permet en outre de représenter des fonctions complexes, telles que les fonctions mathématiques en les mettant directement en liaison avec ces pavés logiques.

Le langage de programmation LOG fait partie du logiciel de base STEP 7.

Exemple :

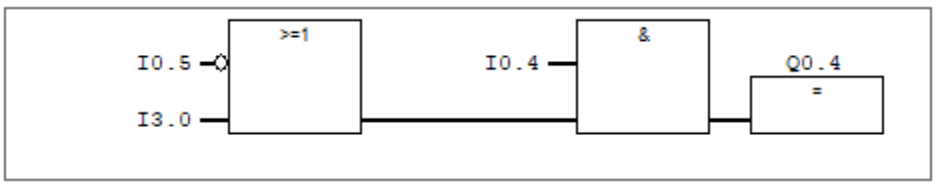


Figure 3.12 : Exemple de programme Logigramme

Dans le langage LOG, vous créez le programme en utilisant un éditeur incrémental.

3.5.1.4 Langage SCL

Le langage de programmation SCL (Structured Control Language) optionnel est un langage évolué textuel, dont la structure de la langue correspond pour l'essentiel à la norme CEI 1131-3. Grâce à ses instructions en langage évolué et contrairement au langage LIST, ce langage proche du PASCAL simplifie entre autres la programmation de boucles et de branches conditionnelles. SCL est de ce fait tout particulièrement adapté au calcul de formules, aux algorithmes d'optimisation complexes ou à la gestion de grandes quantités de données.

Dans le langage SCL, vous créez le programme dans une source SCL, en utilisant un éditeur orienté source.

Exemple :

```
FUNCTION CARRE : INT

 VAR_INPUT
  valeur : INT;
 END_VAR

 BEGIN
   IF valeur <= 181 THEN
       CARRE := valeur * valeur; //Calcul de la valeur de la fonction
   ELSE
       CARRE := 32_767; // Fournir la valeur maximale en cas de

   END_IF;
END_FUNCTION
```

Figure 3.13 : Exemple de programme SCL

3.5.1.5 Langage GRAPH (Grafcet)

C'est un langage graphique utilisé pour décrire les opérations séquentielles. Le procédé est représenté comme une suite connue d'étapes (états stables), reliées entre elles par des transitions, une condition booléenne est attachée à chaque transition. Les actions dans les étapes sont décrites avec les langages IL, LD ou FBD.

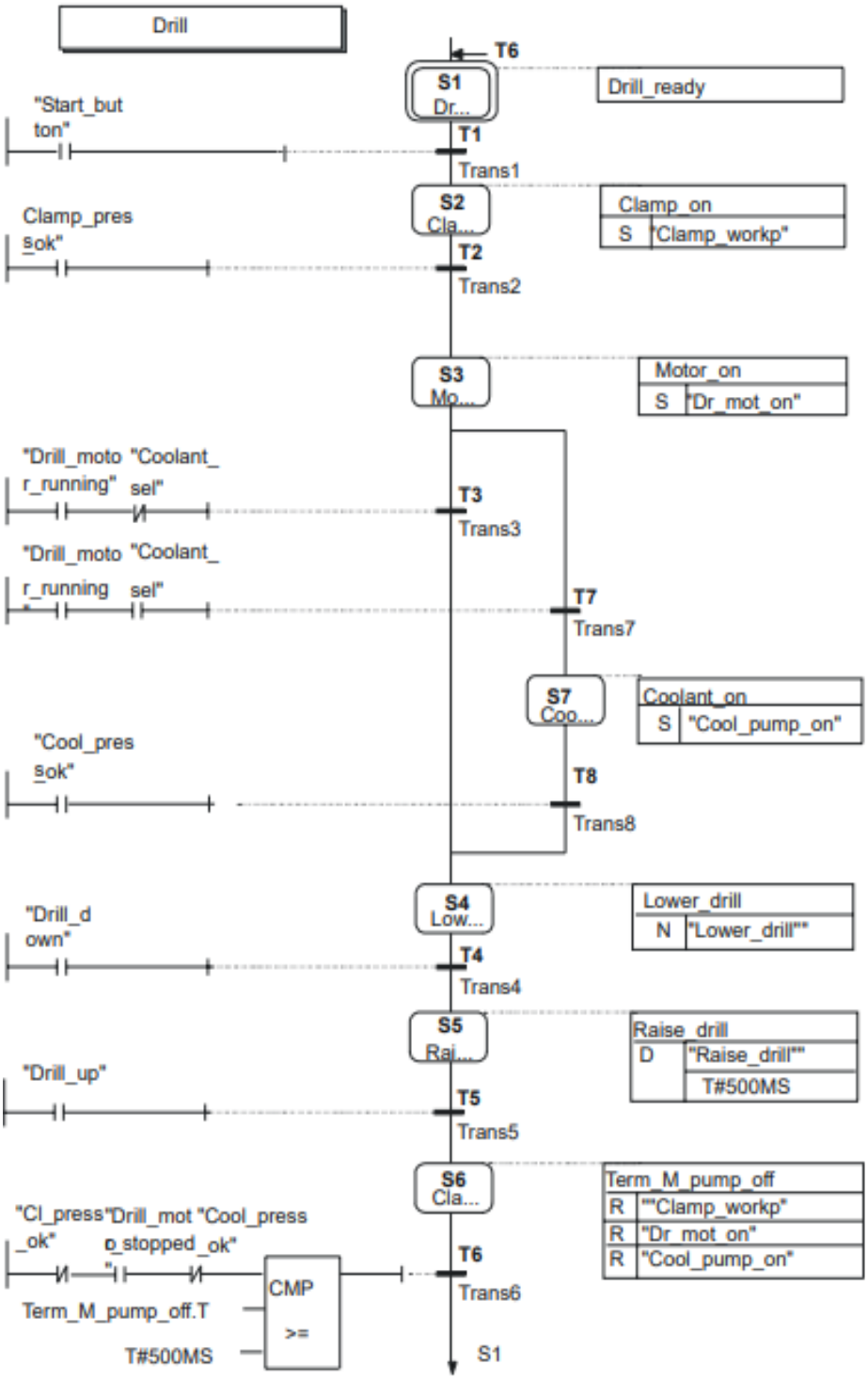


Figure 3.14 : Exemple de programme Grafcet

3.5.2 Structure général d'un programme STEP 7

La structure générale d'un programme STEP 7 est organisée comme suit (Figure 3.5)

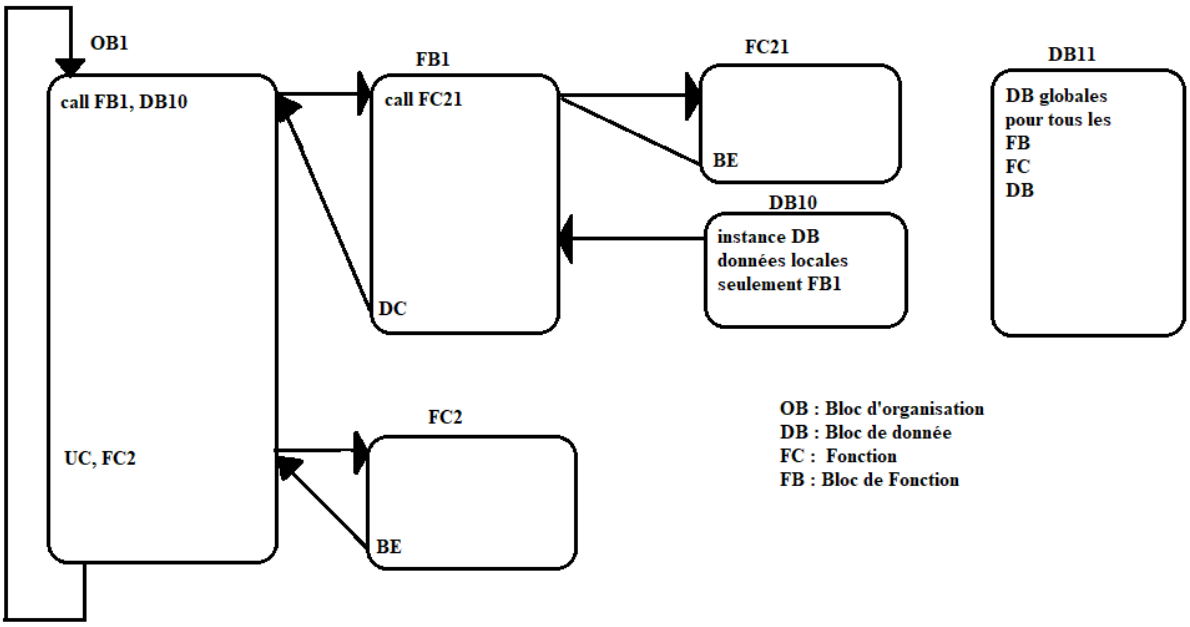


Figure 3.15 : Exemple sur la structure générale d'un programme STEP 7

Les blocs de code sont les blocs du programme utilisateur qui contiennent les instructions à exécuter. Il existe les blocs de code suivants :

Blocs d'organisation OB

Le traitement de programme cyclique constitue le traitement normal pour les automates programmables. Le système d'exploitation appelle l'OB1 cycliquement et déclenche ainsi le traitement cyclique du programme utilisateur.

Blocs de fonction FB

Un bloc fonctionnel contient un programme qui est exécuté quand ce bloc fonctionnel est appelé par un autre bloc de code. Les blocs fonctionnels facilitent la programmation de fonctions complexes souvent utilisées.

Fonctions FC

Les fonctions font partie des blocs que vous programmez vous-même. Une fonction est un bloc de code sans mémoire. Les variables temporaires d'une fonction sont sauvegardées dans la pile des données locales. Ces données sont perdues à l'achèvement de la fonction.

Les fonctions peuvent faire appel à des blocs de données globaux pour la sauvegarde des données. Comme une fonction ne dispose pas de mémoire associée, vous devez toujours indiquer des paramètres effectifs pour elle. Vous ne pouvez pas affecter de valeur initiale aux données locales d'une FC.

Blocs de données DB

Les blocs de données ne contiennent pas d'instructions STEP 7. Ils servent à l'enregistrement des données utilisateur. Ils contiennent des données variables que le programme utilisateur utilise. Les blocs de données globaux servent à l'enregistrement de données utilisateur pouvant être utilisées par tous les autres blocs [15].

3.5.3 Régulation PID dans STEP7

Régulation continue avec le FB 41 « CONT_C»

Le bloc FB 41 « CONT_C » sert à régler des processus industriels à grandeurs d'entrée et de sortie continues sur les automates programmables SIMATIC S7. Le paramétrage vous permet d'activer ou de désactiver des fonctions partielles du régulateur PID et donc d'adapter ce dernier au système à commander.

Vous pouvez utiliser le régulateur comme régulateur PID de maintien autonome mais aussi comme régulateur en cascade, de mélange ou de rapport dans des régulations à plusieurs boucles. Sa méthode de travail se base sur l'algorithme PID du régulateur à échantillonnage à sortie analogique [16].

Description En plus des fonctions traitant la consigne et la mesure, le FB 41 réalise un régulateur PID (Figure 3.16) prêt à l'emploi avec sortie continue de la grandeur de réglage et possibilité d'influencer à la main la valeur de réglage. Il propose les fonctions partielles suivantes :

- Branche de consigne.
- Branche de mesure.
- Formation du signal d'erreur.
- Algorithme PID.
- Traitement de la valeur de réglage manuelle.
- Traitement de la valeur de réglage.
- Action anticipatrice.

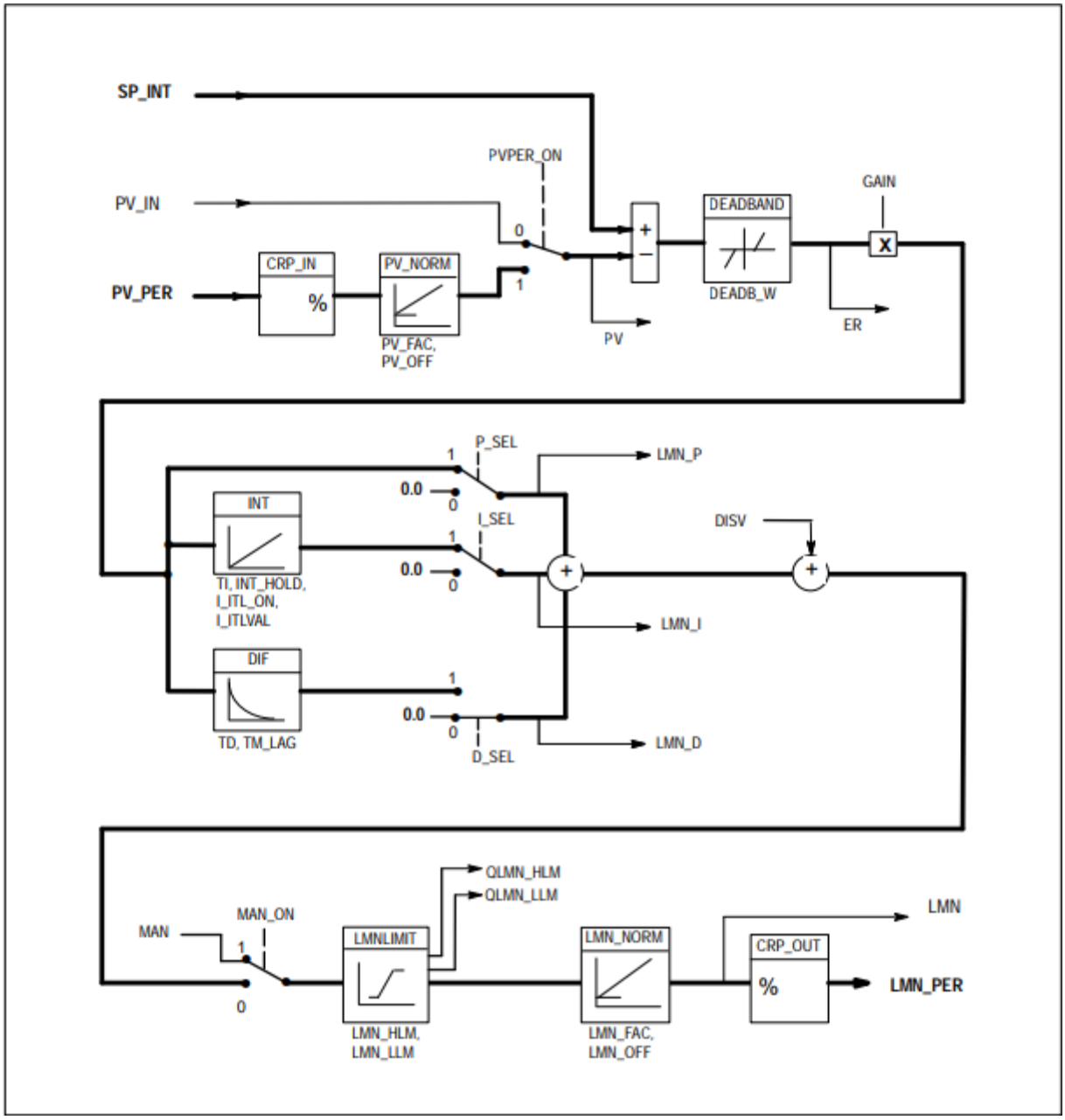


Figure 3.16 : Schéma fonctionnel du PID

Les Tableaux 3.1 et 3.2 résume la signification de chaque variable du PID de la Figure 3.16.

Paramètre	Type de données	Valeurs admises	Par défaut	Description
COM_RST	BOOL		FALSE	COMPLETE RESTART / Démarrage Le bloc renferme un sous-programme de démarrage qui est exécuté quand cette entrée est à 1.
MAN_ON	BOOL		TRUE	MANUAL VALUE ON / Activation du mode manuel Quand cette entrée est à 1, la boucle de régulation est interrompue. La valeur de réglage manuelle est sortie comme grandeur de réglage.
P_SEL	BOOL		TRUE	PROPORTIONAL ACTION ON / Activation de l'action proportionnelle Dans l'algorithme PID, il est possible d'activer et de désactiver séparément chacune des actions. L'action P est active quand cette entrée est à 1.
I_SEL	BOOL		TRUE	INTEGRAL ACTION ON / Activation de l'action par intégration Dans l'algorithme PID, il est possible d'activer et de désactiver séparément chacune des actions. L'action I est active quand cette entrée est à 1.
D_SEL	BOOL		FALSE	DERIVATIVE ACTION ON / Activation de l'action par dérivation Dans l'algorithme PID, il est possible d'activer et de désactiver séparément chacune des actions. L'action D est active quand cette entrée est à 1.
CYCLE	TIME	>= 1 ms	T#1s	SAMPLE TIME / Période d'échantillonnage Le temps s'écoulant entre les appels de bloc doit être constant. Il est indiqué au niveau de cette entrée.
SP_INT	REAL	–100,0 à 100,0 (%) ou grandeur physique	0,0	INTERNAL SETPOINT / Consigne interne Cette entrée sert à introduire une valeur de consigne.
PV_IN	REAL	–100,0 à 100,0 (%) ou grandeur physique	0,0	PROCESS VARIABLE IN / Mesure d'entrée Cette entrée permet de paramétrer une valeur de mise en service ou d'appliquer une mesure externe en virgule flottante.
MAN	REAL	–100,0 à 100,0 (%) ou grandeur physique	0,0	MANUAL VALUE / Valeur de réglage manuelle Cette entrée sert à introduire une valeur de réglage manuelle moyennant des fonctions de contrôle-commande.
GAIN	REAL		2,0	PROPORTIONAL GAIN / Coefficient d'action proportionnelle Cette entrée indique le gain du régulateur.
TI	TIME	>= CYCLE	T#20 s	RESET TIME / Temps d'intégration Cette entrée détermine la réponse temporelle de l'intégrateur
TD	TIME	>= CYCLE	T#10 s	DERIVATIVE TIME / Temps de dérivation Cette entrée détermine la réponse temporelle de l'unité de dérivation.
LMN_HLM	REAL	LMN_LLM à 100,0 (%) ou grandeur physique	100,0	MANIPULATED VALUE HIGH LIMIT / Limite supérieure de la valeur de réglage La valeur de réglage est toujours bornée à une limite supérieure et une limite inférieure. Cette entrée indique la limite supérieure.
LMN_LLM	REAL	–100,0 à LMN_HLM (%) ou grandeur physique	0,0	MANIPULATED VALUE LOW LIMIT / Limite inférieure de la valeur de réglage La valeur de réglage est toujours bornée à une limite supérieure et une limite inférieure. Cette entrée indique la limite inférieure.

Tableau 3.1 : Paramètres d'entrée (INPUT) du bloc FB 41 « C ONT_C »

Paramètre	Type de données	Valeurs admises	Par défaut	Description
LMN	REAL		0,0	MANIPULATED VALUE / Valeur de réglage Cette sortie donne en virgule flottante la valeur de réglage agissant réellement.
ER	REAL		0,0	ERROR SIGNAL / Signal d'erreur Cette sortie donne le signal d'erreur effectif.

Tableau 3.2 : Paramètres de sortie (OUTPUT) du bloc FB 41 « C ONT_C »

3.5.4 Simulateur S7-PLCSIM

Dans S7-PLCSIM (Figure 3.17), vous pouvez exécuter votre programme utilisateur STEP 7 et l'essayer dans un automate programmable simulé. Cette simulation s'exécute sur votre PC ou console de programmation, une Field PG par exemple. La simulation étant réalisée entièrement dans le logiciel STEP 7, vous n'avez pas besoin de matériel S7 (CPU ou modules de signaux). Avec S7-PLCSIM, vous pouvez simuler des programmes utilisateur STEP 7 qui ont été développés pour les automates S7-300, S7-400 et WinCC flexible.

S7-PLCSIM offre une interface simple au programme utilisateur STEP 7 servant à visualiser et à modifier différents objets tels que les variables d'entrée et de sortie. Pendant que votre programme est traité par la CPU simulée, vous pouvez recourir au logiciel STEP 7. Par exemple, vous pouvez visualiser et forcer des variables avec la table des variables (VAT). S7-PLCSIM offre une interface utilisateur graphique permettant de visualiser et de modifier les variables des programmes d'automatisation, d'exécuter en mode cyclique ou automatique le programme du système cible simulé ou de modifier l'état de fonctionnement de l'automate simulé.

S7-PLCSIM comprend également un objet COM appelé S7ProSim pour accéder par programme à un système cible simulé. S7ProSim vous permet d'écrire du logiciel pour exécuter des actions comme la commutation de la position du commutateur à clé sur le système cible simulé, l'exécution cyclique du programme, la lecture ou l'écriture des valeurs de l'automate, etc. [19]

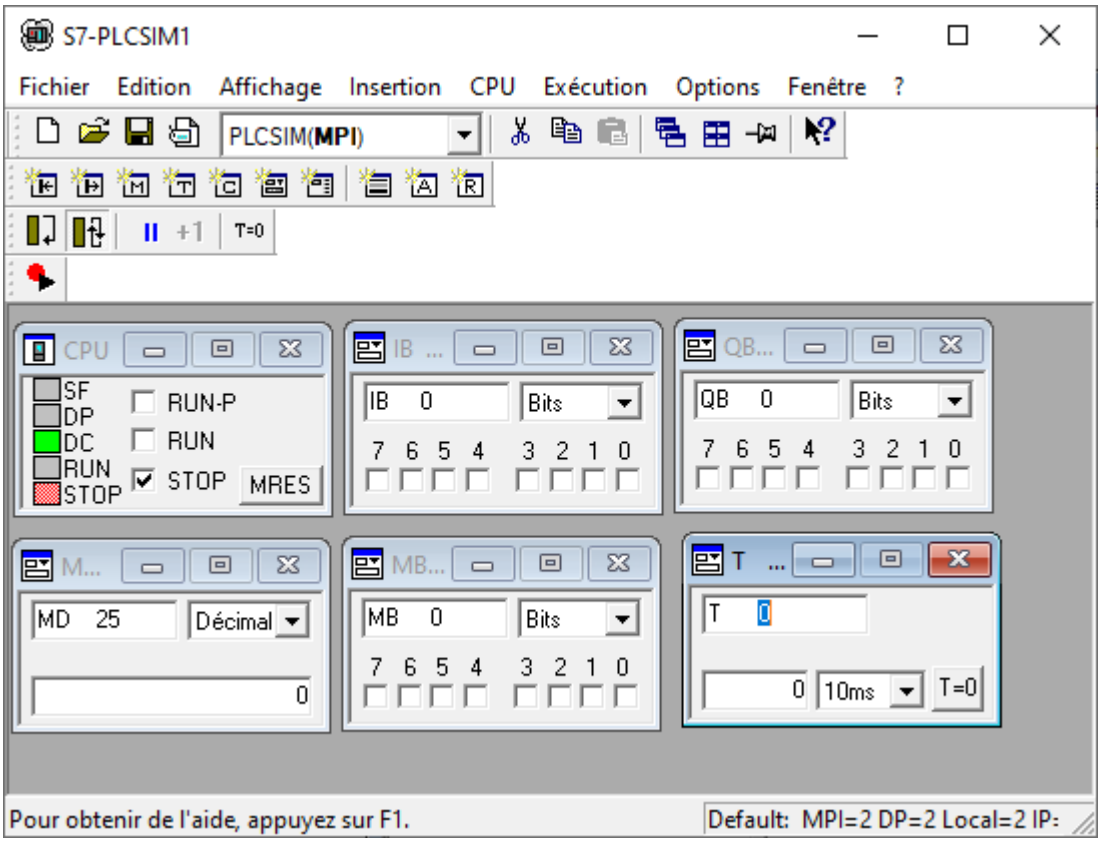


Figure 3.17 : Simulateur S7-PLCSIM

3.6 Présentation de Win CC flexible

3.6.1 Introduction

Lorsque la complexité des processus augmente et que les machines et installations doivent répondre à des spécifications de fonctionnalité toujours plus sévères, l'opérateur a besoin d'un maximum de transparence. Cette transparence s'obtient au moyen de l'Interface Homme-Machine (IHM). Un système IHM constitue l'interface entre l'homme (opérateur) et le processus (machine/installation). Le contrôle proprement dit du processus est assuré par le système d'automatisation. Il existe par conséquent une interface entre l'opérateur et WinCC flexible (sur le pupitre opérateur) et une interface entre WinCC flexible et le système d'automatisation [17].

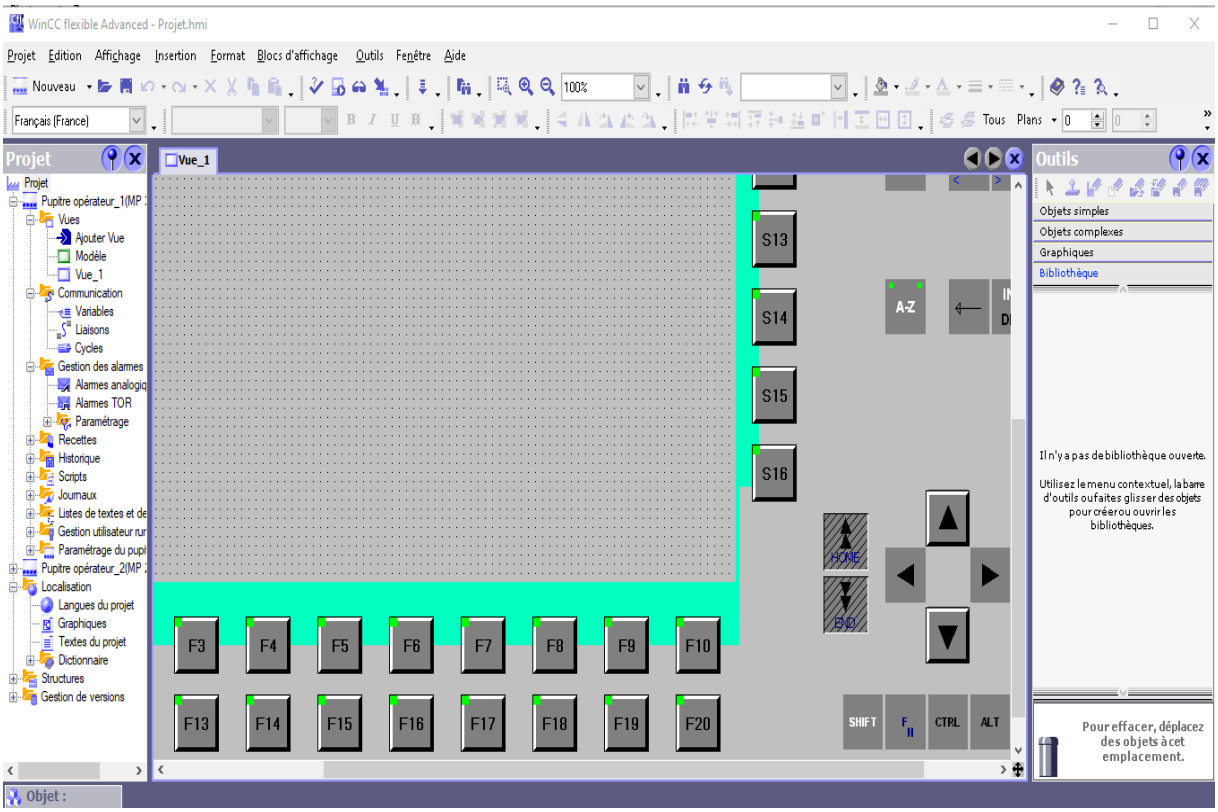


Figure 3.18 : Fenêtre d'accueil WinCC flexible

3.6.2 Tâches d'un système IHM

- Représentation du processus.
- Commande du processus.
- Vue des alarmes.
- Archivage des valeurs processus et d'alarmes.
- Gestion des paramètres du processus. [17]

3.6.3 SIMATIC WinCC flexible

WinCC flexible est le logiciel IHM pour la réalisation, par des moyens d'ingénierie simples et efficaces, de concepts d'automatisation évolutifs, au niveau machine. WinCC flexible réunit les avantages suivants :

- Simplicité.
- Ouverture.
- Flexibilité. [18]

3.6.4 Element de winCC flexible

WinCC flexible Engineering System

WinCC flexible Engineering System est le logiciel avec lequel vous réalisez toutes les tâches de configuration requises. L'édition WinCC flexible détermine les pupitres opérateurs de la gamme SIMATIC HMI pouvant être configurés.

WinCC flexible Runtime

WinCC flexible Runtime est le logiciel de visualisation de process. Dans Runtime, vous exécutez le projet en mode process.

Options WinCC flexible

Les options WinCC flexible permettent d'étendre les fonctionnalités de base de WinCC flexible. Chaque option nécessite une licence particulière [18].

3.6.5 Intégration de WinCC flexible à STEP7

Pour intégrer un projet WinCC existant dans un projet STEP 7, procédez comme suit :

1. Ouvrez la configuration WinCC flexible.
2. Sélectionnez le menu "Projet > Intégrer dans le projet STEP 7..." Le dialogue "Intégrer dans les projets STEP 7" s'ouvre.
3. Sélectionnez dans le dialogue le projet STEP 7 correspondant.

Si le projet souhaité ne se trouve pas dans la liste, naviguez par le champ "Rechercher dans" vers le dossier qui contient le projet STEP 7.

Après la sélection du projet STEP 7, l'intégration est exécutée

3.7 Conclusion

Dans ce chapitre, nous avons présenté les différents composants des automates programmables industriels de la gamme S7 et leurs caractéristiques, ainsi que les logiciels de programmation Step 7 et de supervision WinCC flexible.

Le choix d'un API est lié à l'environnement. Plus ce dernier est perturbé, plus les exigences en termes de sûreté de fonctionnement sont grandes, plus l'API s'impose face à des solutions concurrentes.

CHAPITRE 4

Régulation et supervision du niveau d'un réservoir cylindrique

4.1 Introduction

Dans ce chapitre nous présentons le système à réservoir cylindrique de Factory I/O, le programme utilisé pour la régulation du niveau avec sa partie IHM. A la fin du chapitre, des résultats de simulations seront présentés.

4.2 Présentation du système à réservoir

4.2.1 Factory I/O

Factory I/O est un logiciel de simulation 3D permettant de concevoir et de simuler des systèmes industriels complets. Développé dans le même esprit que les jeux vidéo actuels, Factory I/O offre un haut degré de réalisme pendant les phases de conception et de simulation. Le logiciel peut s'interfacer avec des automates réels permettant d'effectuer le pilotage depuis l'extérieur.

4.2.2 Interface du logiciel

L'interface du logiciel est assez simple et dispose de 3 menus et plusieurs boutons permettant de gérer les caméras, d'afficher la palette, d'afficher les capteurs, d'afficher les actionneurs, d'effectuer une simulation etc…

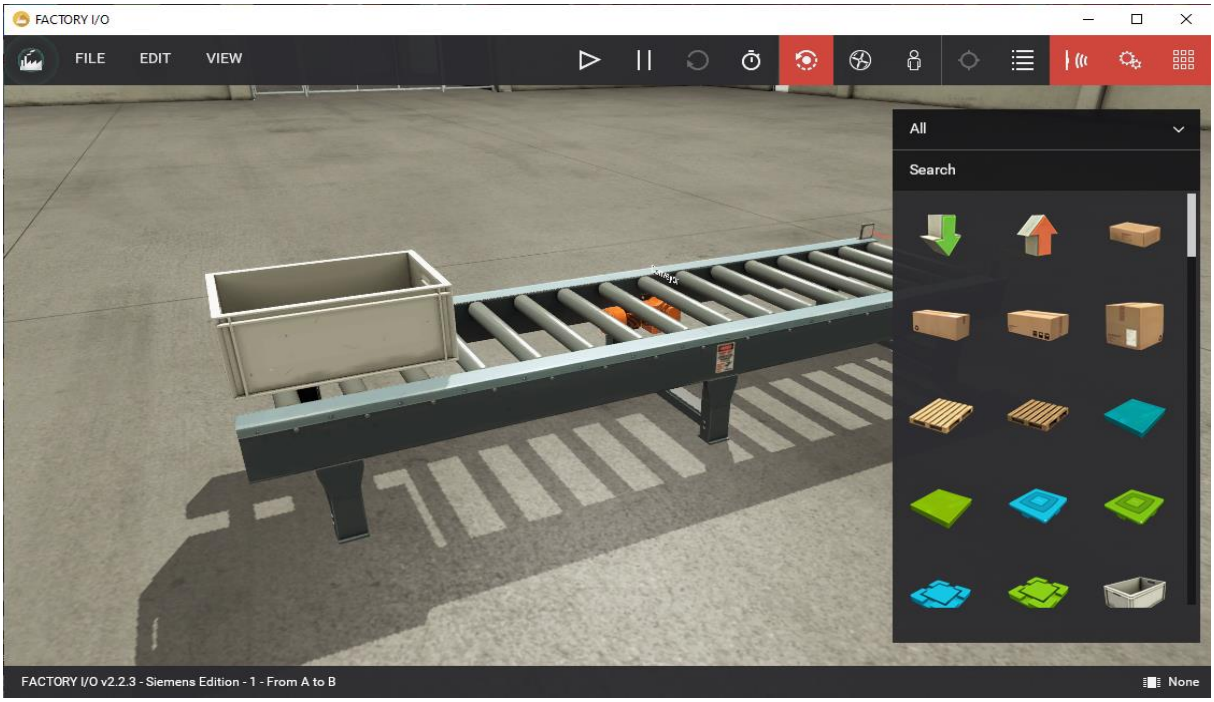


Figure 4.1 : L'interface du logiciel Factory I/O

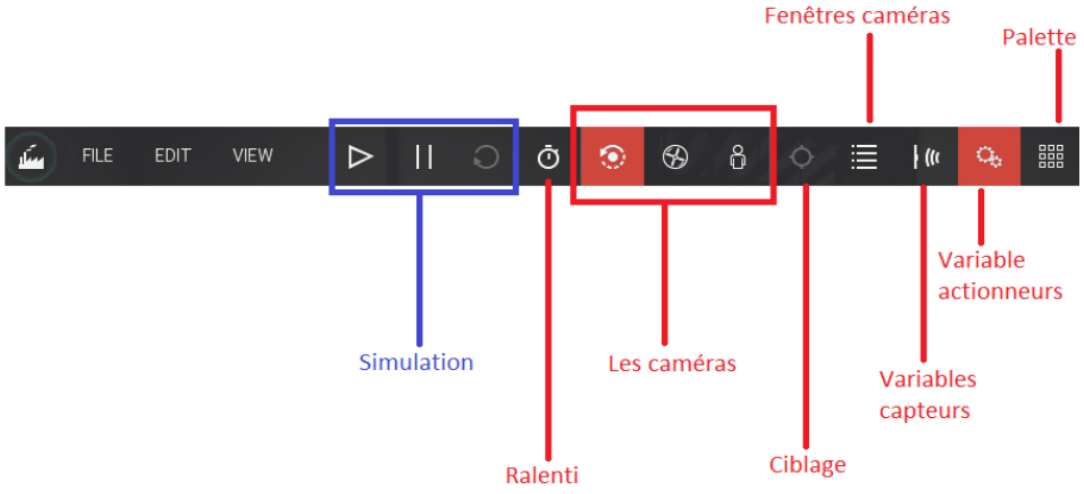


Figure 4.2 : La barre des tâches

Caméras

Il y'a principalement 3 caméras :

- Le Fly camera.
- L'Orbite camera.
- Le first Person camera.

Pour manipuler ces différentes caméras, vous aurez à utiliser les deux boutons de votre souris, la molette et les touches directions de votre clavier. Pour être à l'aise avec les différentes caméras, vous devrez vous exercer.

Palettes

Au niveau de la palette se trouve les différents composants permettant de concevoir brique par brique votre système automatisé. Au niveau de la palette on peut retrouver des tapis roulants, des boutons poussoirs, des voyants etc...

Vous pouvez concevoir plusieurs types de systèmes en faisant un simple glissé-déposer du composant dans l'interface d'édition.

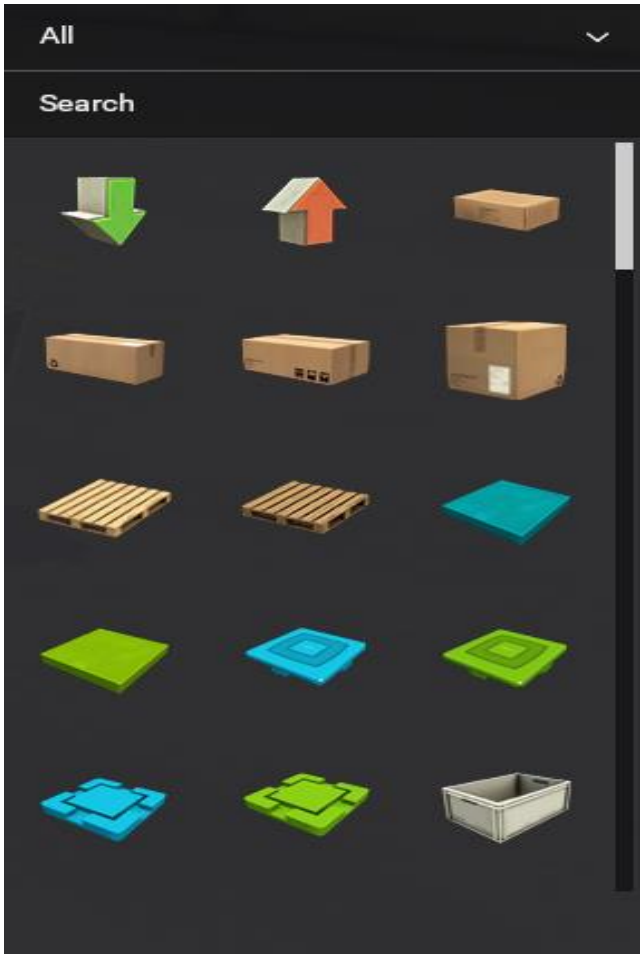


Figure 4.3 : La palette

Scènes prédéfinies

Vous pouvez vous aider des scènes prédéfinis qui sont des systèmes déjà conçus qui vous permettront de découvrir comment fonctionne le logiciel. Pour afficher les scènes préfinis, cliquez sur « File », puis sur « Open ».

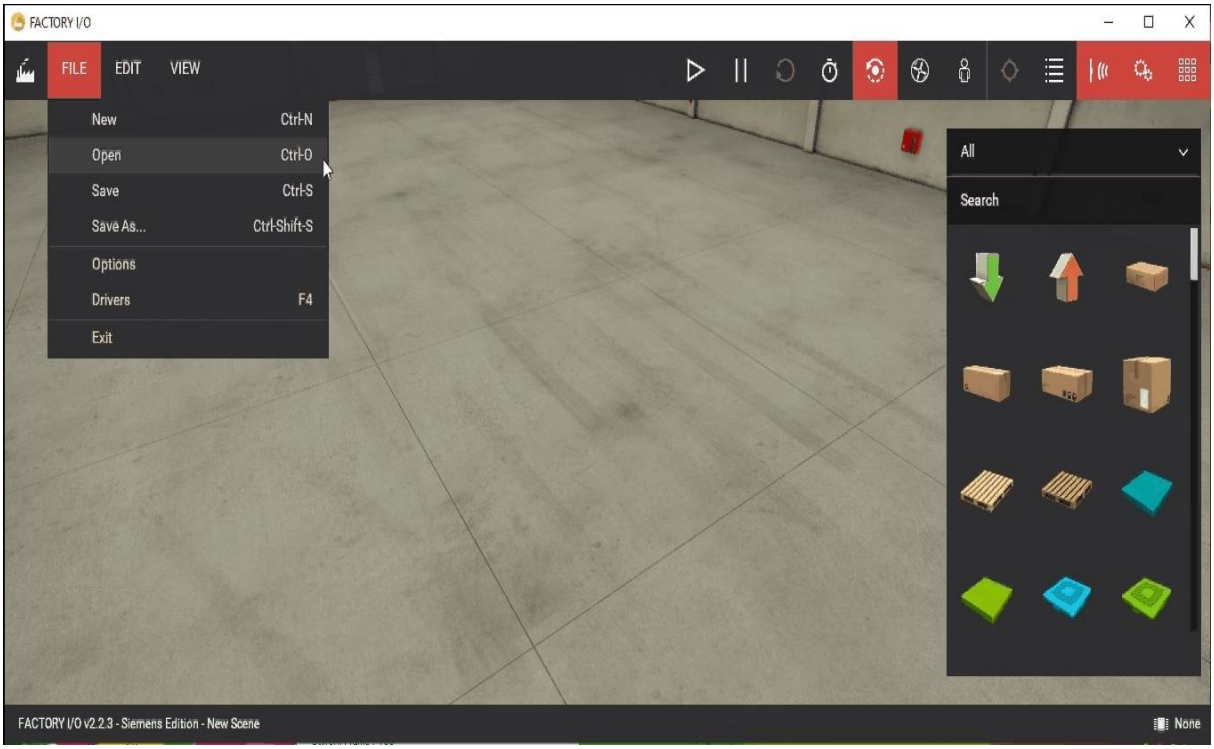


Figure 4.4 : Ouverture d'une scène

Vous pourrez ainsi choisir une scène à afficher parmi la liste des scènes prédéfinies.

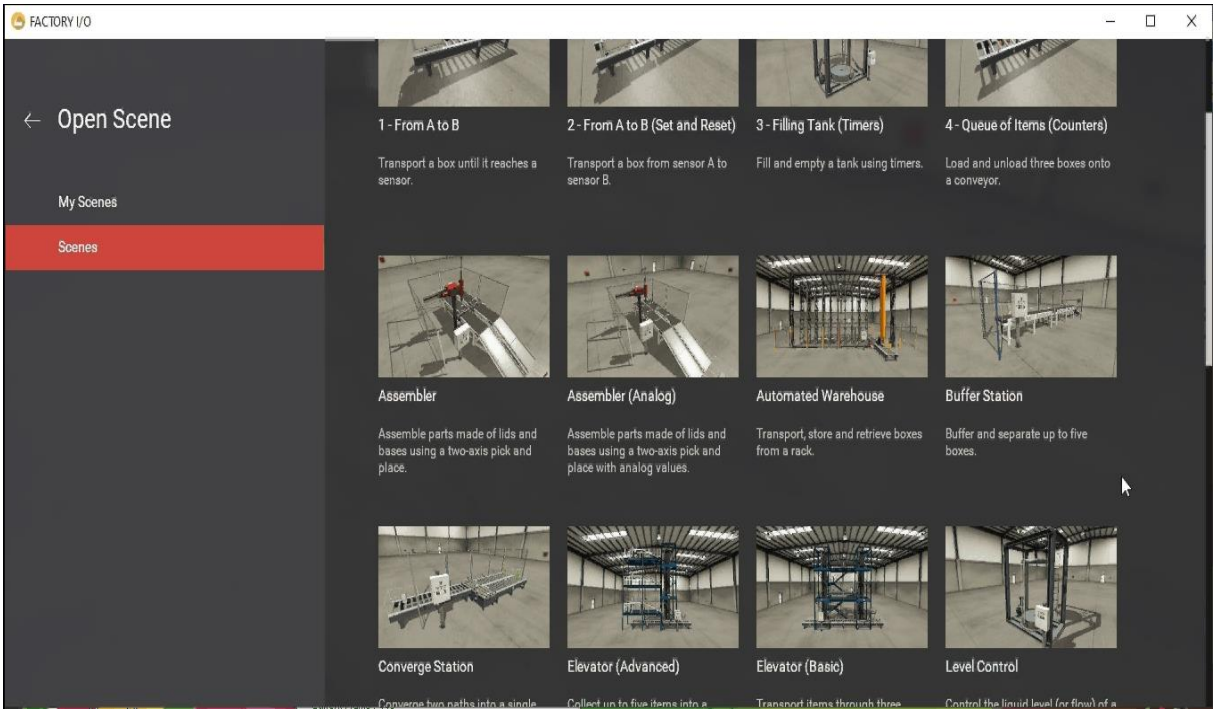


Figure 4.5 : Liste des scènes prédéfinies

Drivers

Les drivers permettent de contrôler le système automatisé conçu via l'interface d'édition. Il existe plusieurs types de drivers :

Les drivers pour les automates Siemens, Schneider, Rockwell etc... , aussi le driver Connect I/O qui est l'automate logiciel intégré à Factory I/O.

Pour activer un driver donné, cliquez sur FILE >> DRIVERS

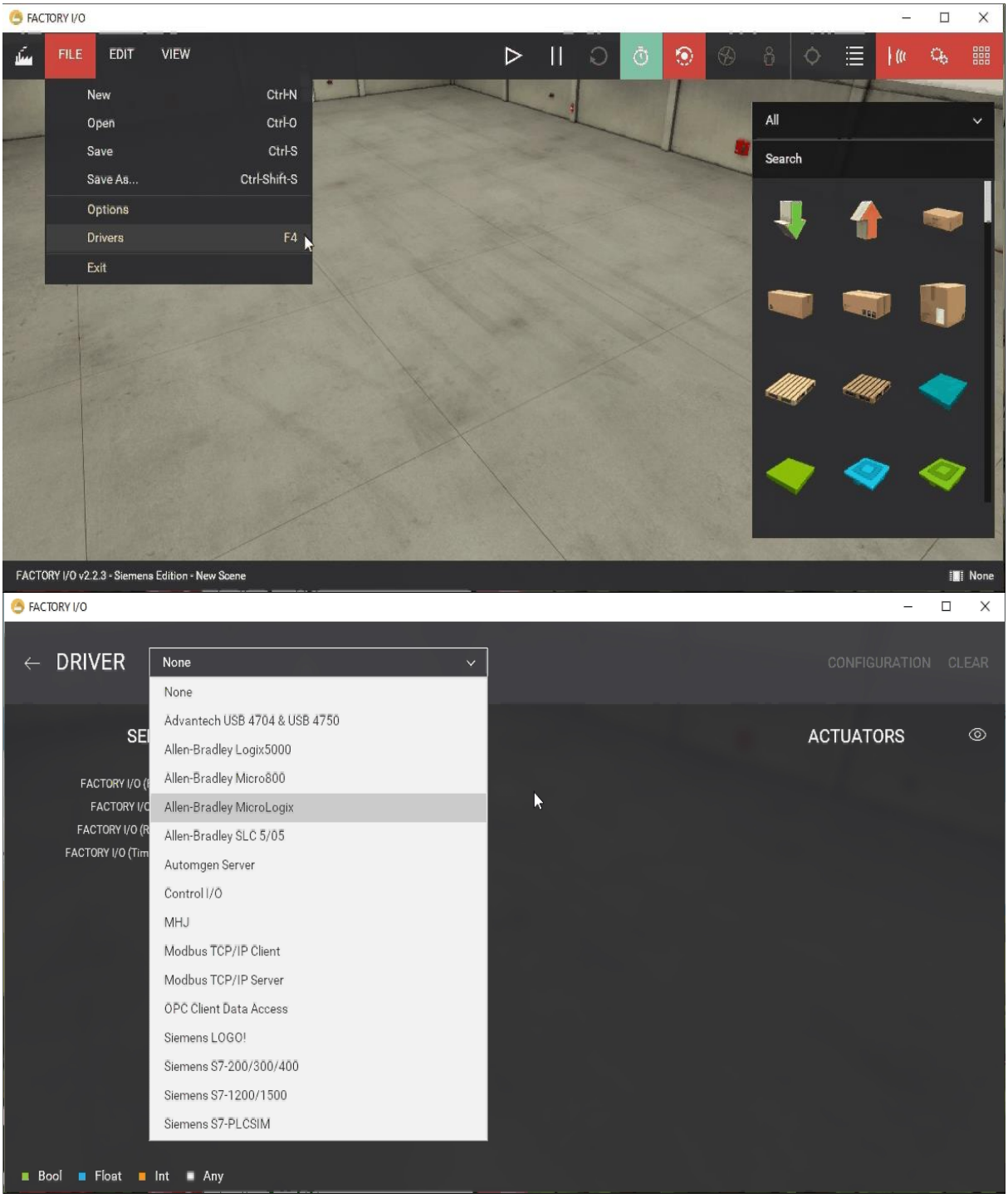


Figure 4.6 : types des drivers

4.2.3 Système à réservoir

Ouverture de la scène «Level Control »

Pour ouvrir la scène prédéfinie «Level Control », cliquez sur FILE >> OPEN >> SCENES >> Level Control

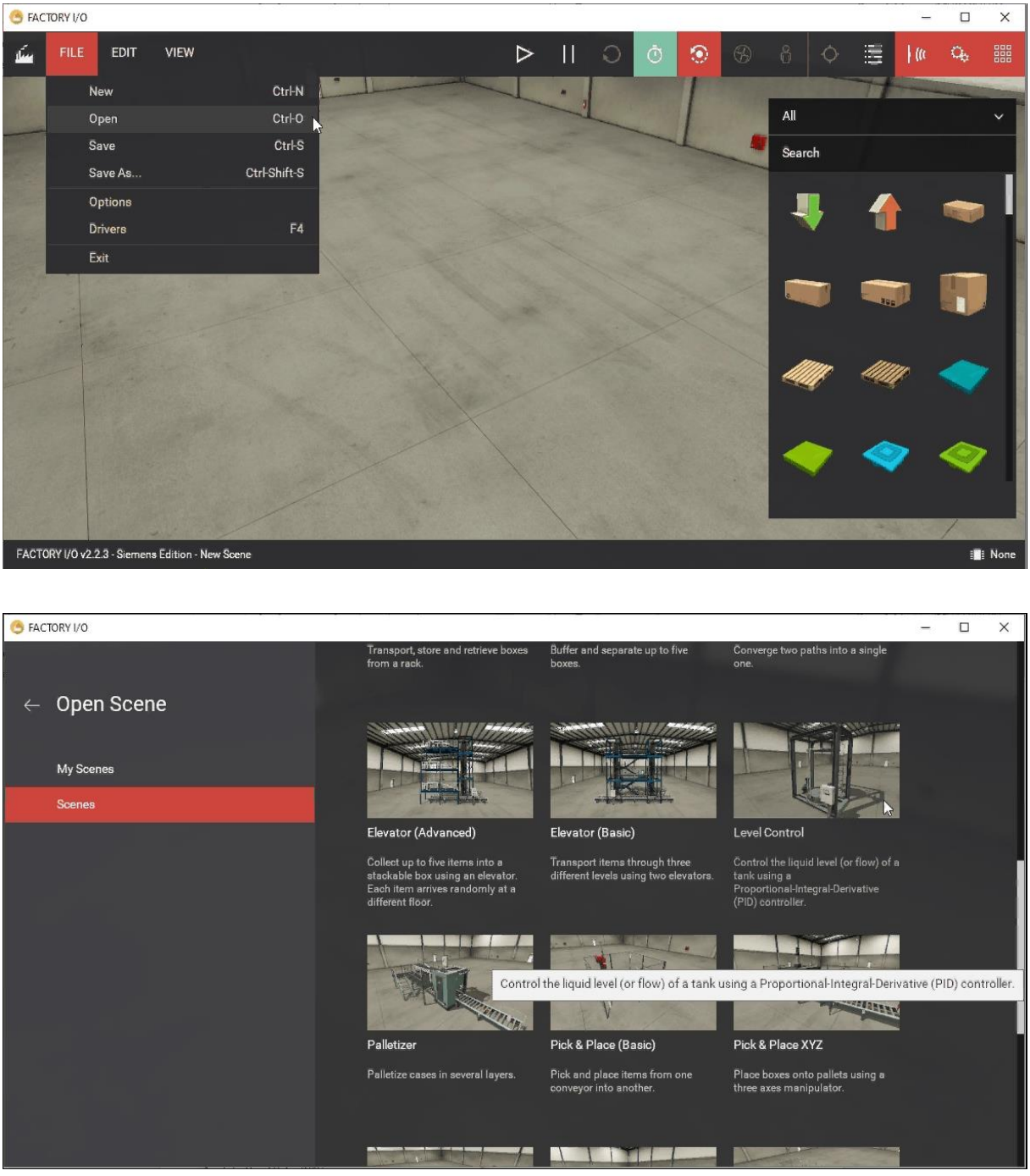


Figure 4.7 : Ouverture de la scène Level Control

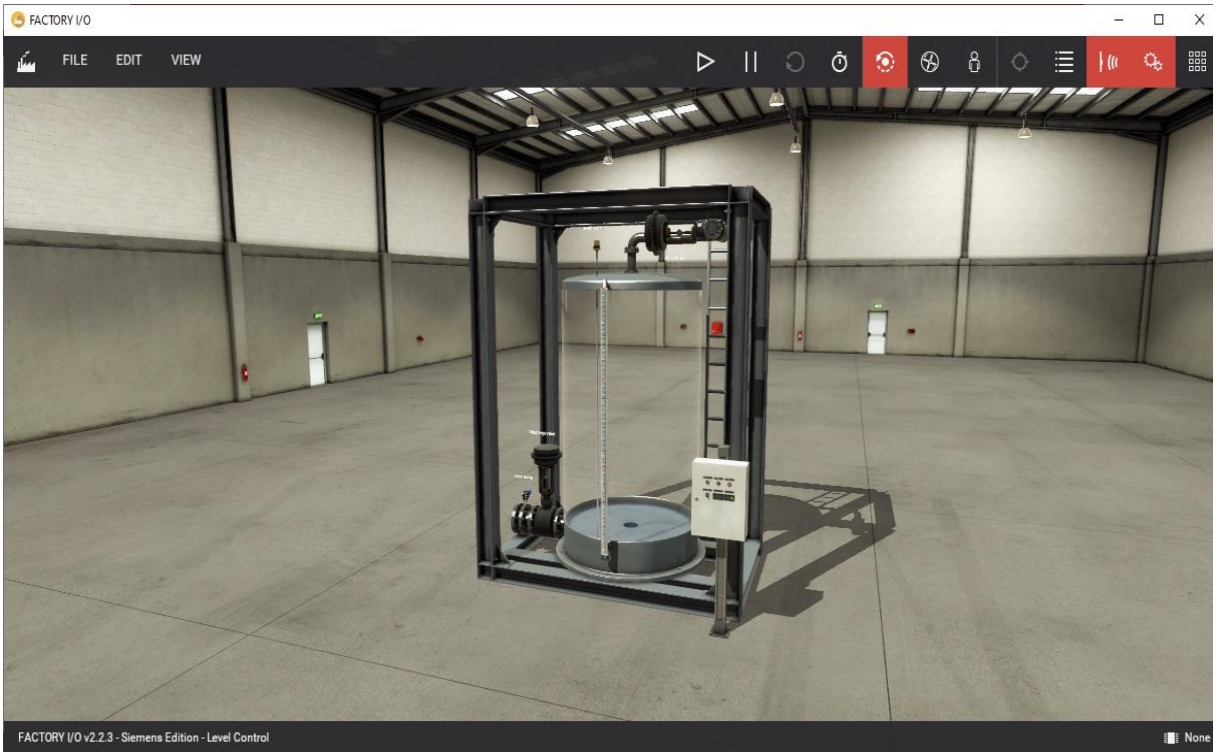
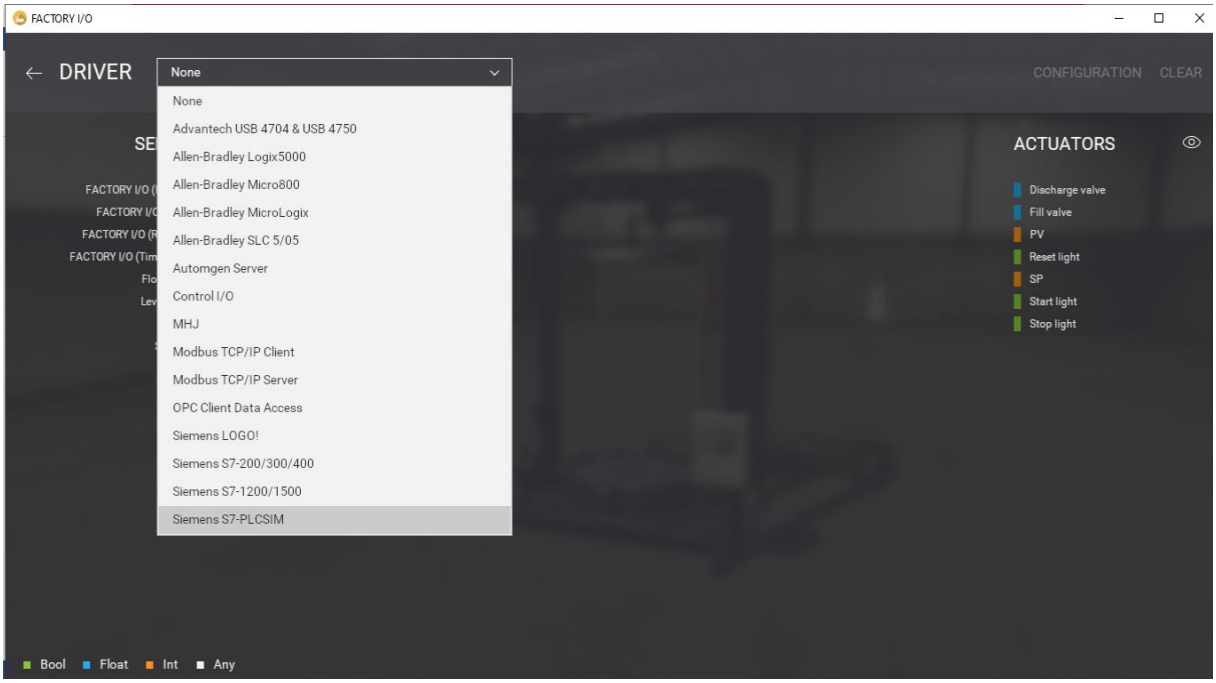


Figure 4.8 : Exemple d'une scène (control the liquid level)

Connections avec STEP 7

Pour faire la connections entre le STEP 7 et le Factory I/O il faut d'abord lancer la simulation sous STEP 7 après dans le Factory I/O sélectionner le driver Siemens S7-PLCSIM, cliquez sur connect.



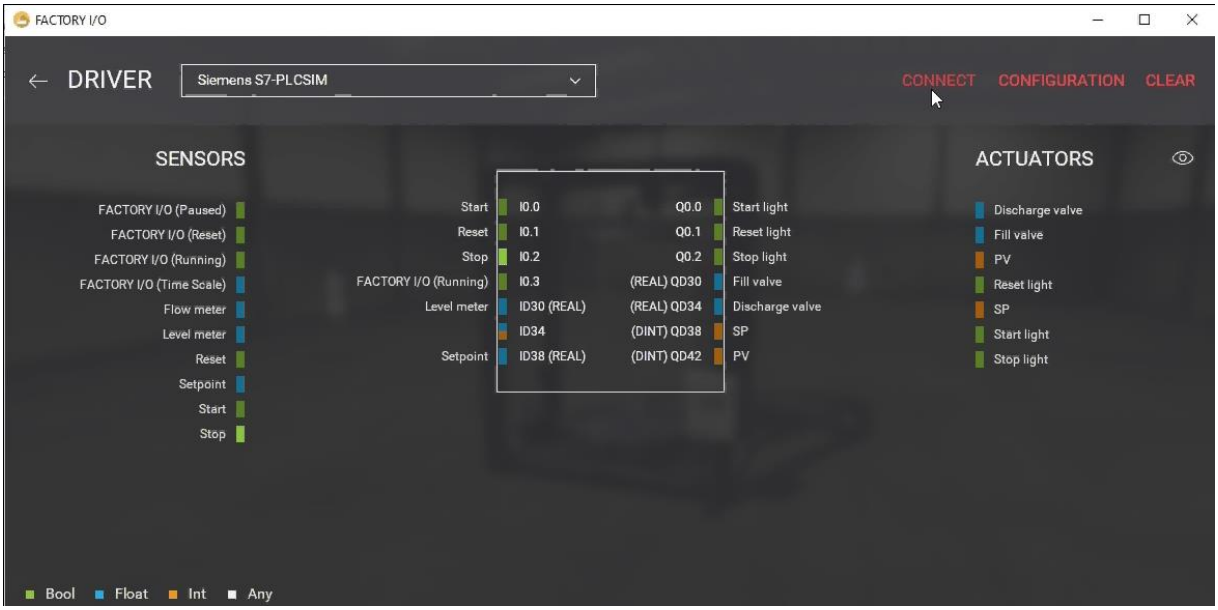


Figure 4.9 : Connections avec PLCSIM

Actionneurs et capteurs du système

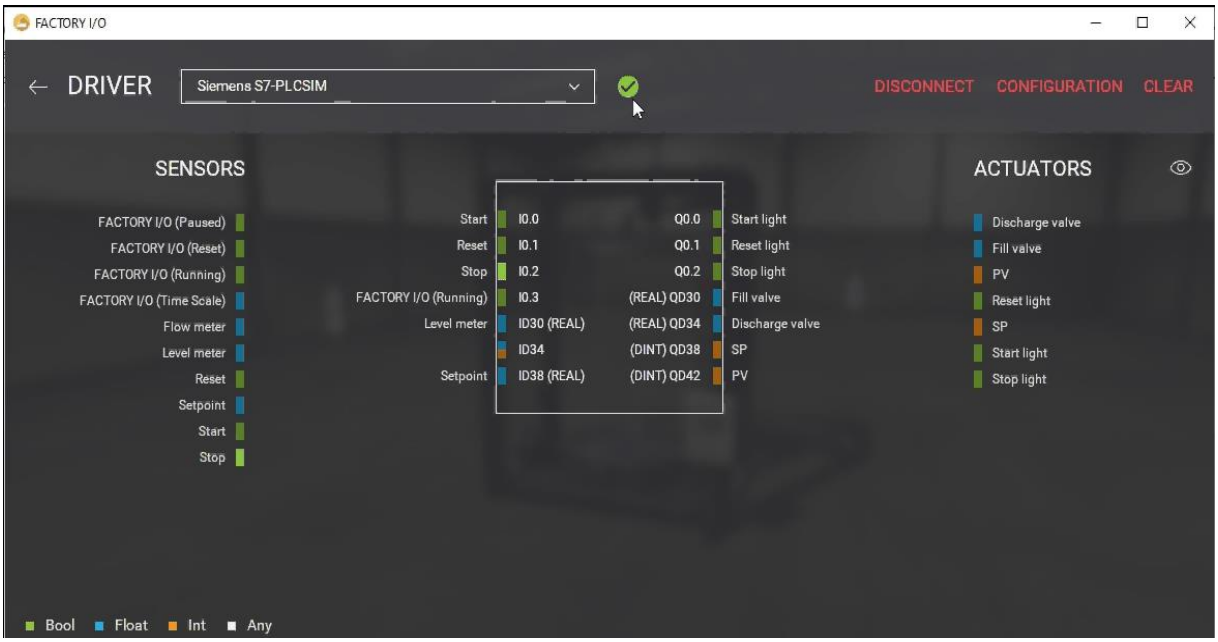


Figure 4.10 : Actionneurs et capteurs du système

Les actionneurs : Il y a deux actionneurs qui sont :

- la vanne de remplissage «Fill valve ».
- la vanne de décharge «Discharge valve ».

Les capteurs : Il y a un seul capteur :

- capteur de niveau «Level meter ».

Entrées/sorties du système

Les entrées/sorties logiques :

Les entrées sont :

- Le bouton départ « Start».
- Le bouton reset « Reset».
- Le bouton stop « Stop ».
- L'état du processus « FACTORY I/O (Running) ».

Les sorties sont :

- La lampe départ « Start light ».
- La lampe reset « Reset light ».
- La lampe stop « Stop light ».

Les entrées/sorties analogiques

Les entrées :

- la consigne «Setpoint ».

Les sorties :

- la vanne de remplissage « Fill valve ».
- la vanne de décharge « Discharge valve ».
- la consigne « SP » et la valeur du processus « PV ».

4.3 Présentation de la partie commande

4.3.1 Bloc d'organisation (OB1+OB35)

Un OB est appelé cycliquement par le système d'exploitation et réalise ainsi l'interface entre le programme utilisateur et le système d'exploitation. Le dispositif de commande est informé dans cet OB par des commandes d'appel de blocs, de quels blocs de programme il doit traiter. Le bloc OB1 contient le programme principal et le bloc OB35 contient le régulateur PID (FB 41). Le bloc OB 35 est appelé cycliquement à chaque période d'échantillonnage.

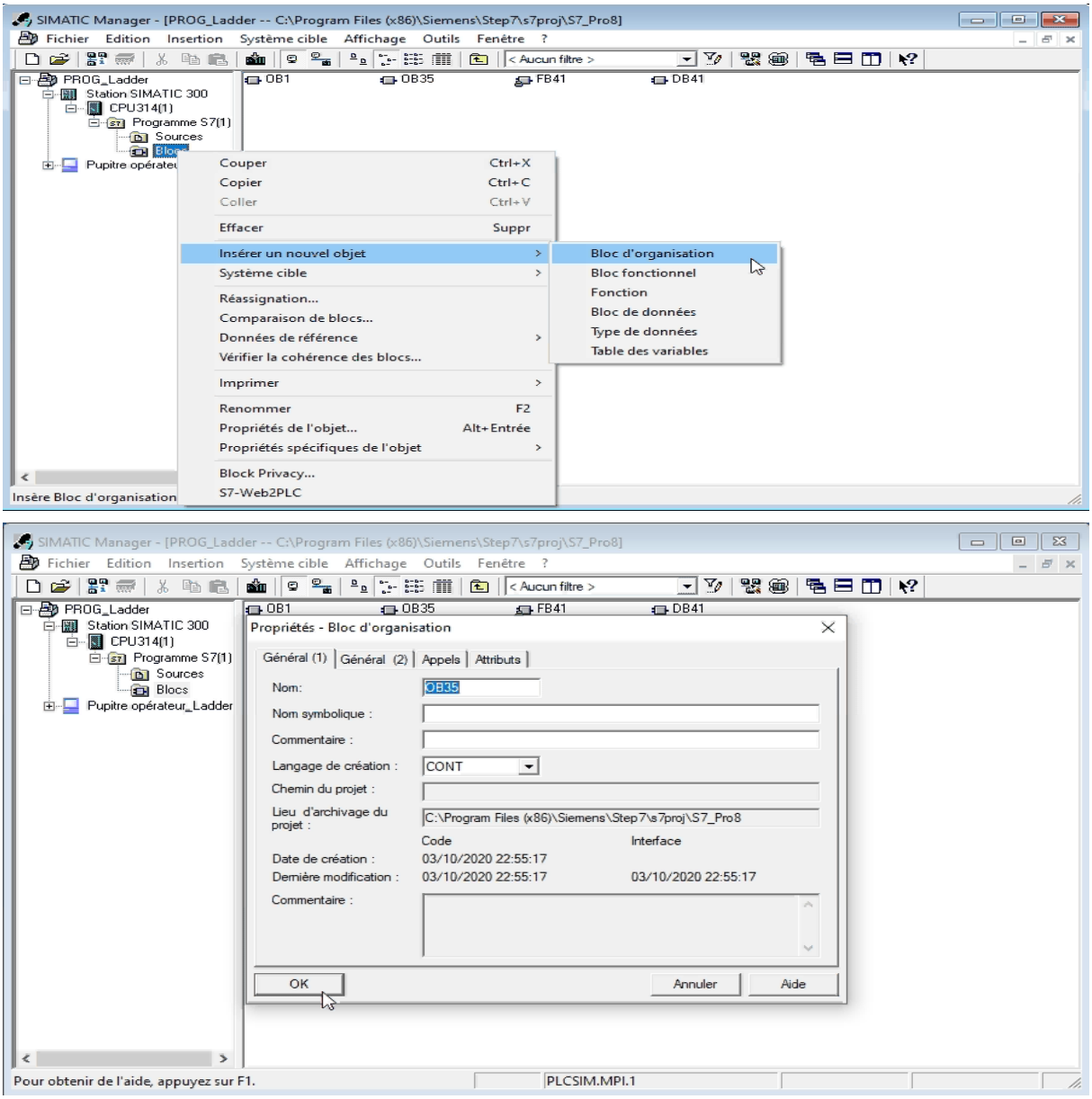
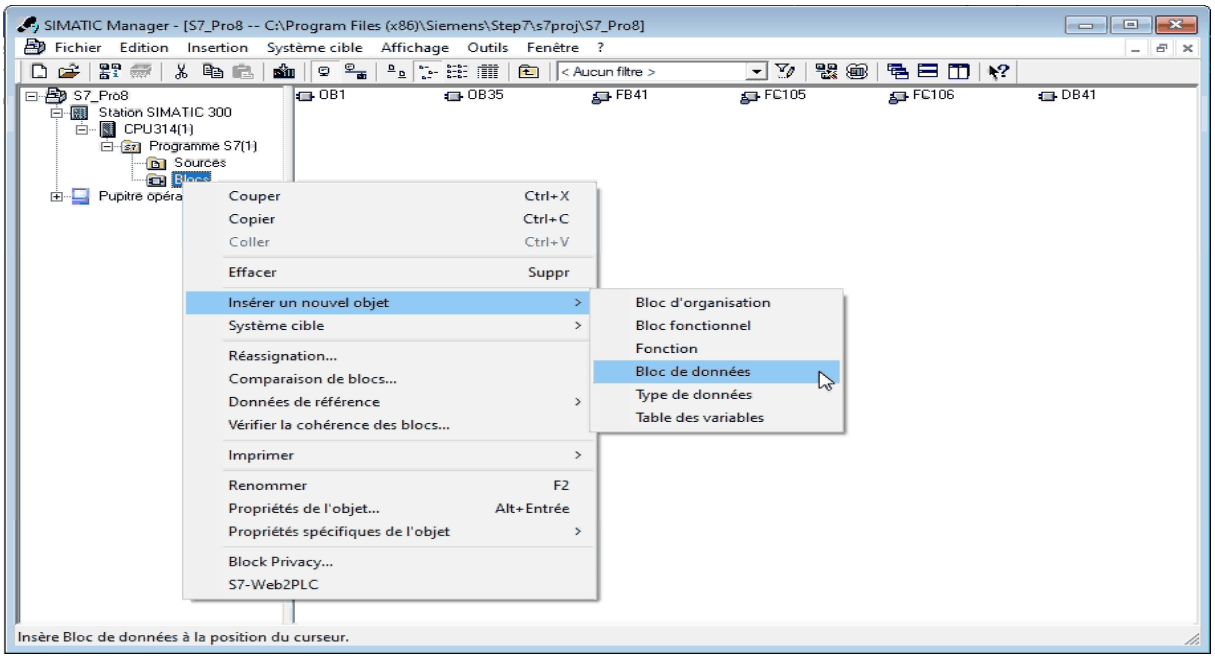


Figure 4.11 : Création du bloc d'organisation

4.3.2 Bloc de données DB

Les DB sont employés afin de tenir à disposition de l'espace mémoire pour les variables de données. Il y a deux catégories de blocs de données. Les DB globaux où tous les OB, FB et FC peuvent lire des données enregistrées et écrire eux-mêmes des données dans le DB. Les instances DB sont attribuées à un FB défini.

Création et contenu d'un DB



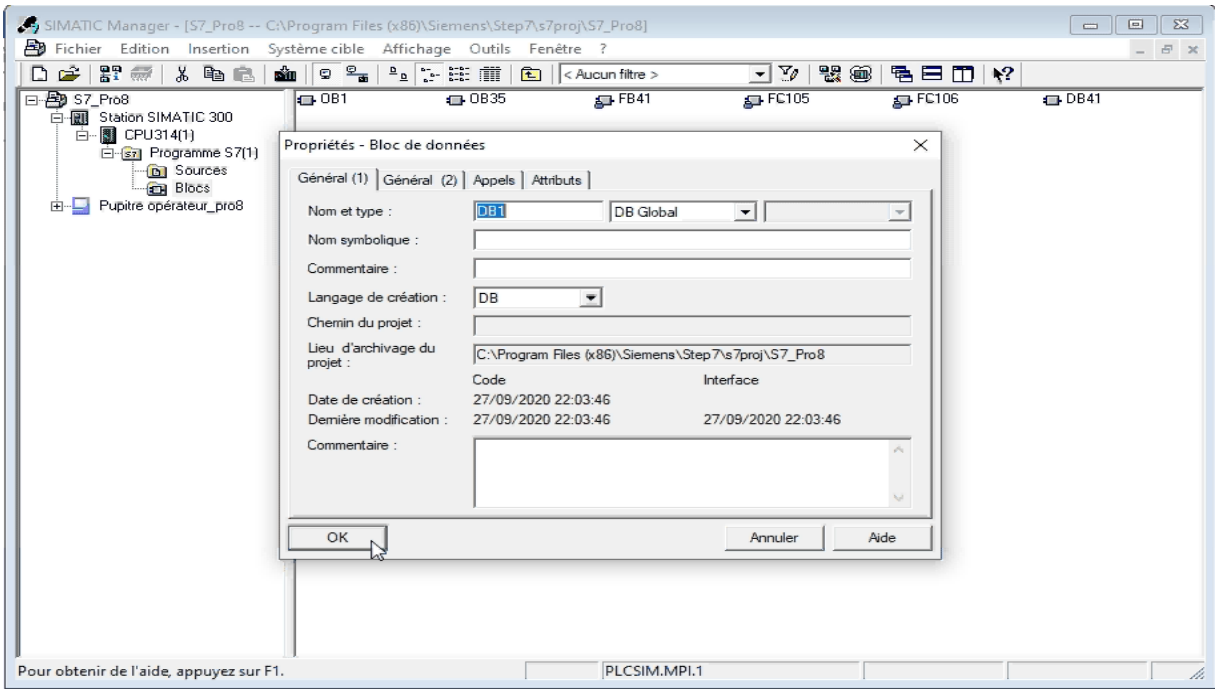
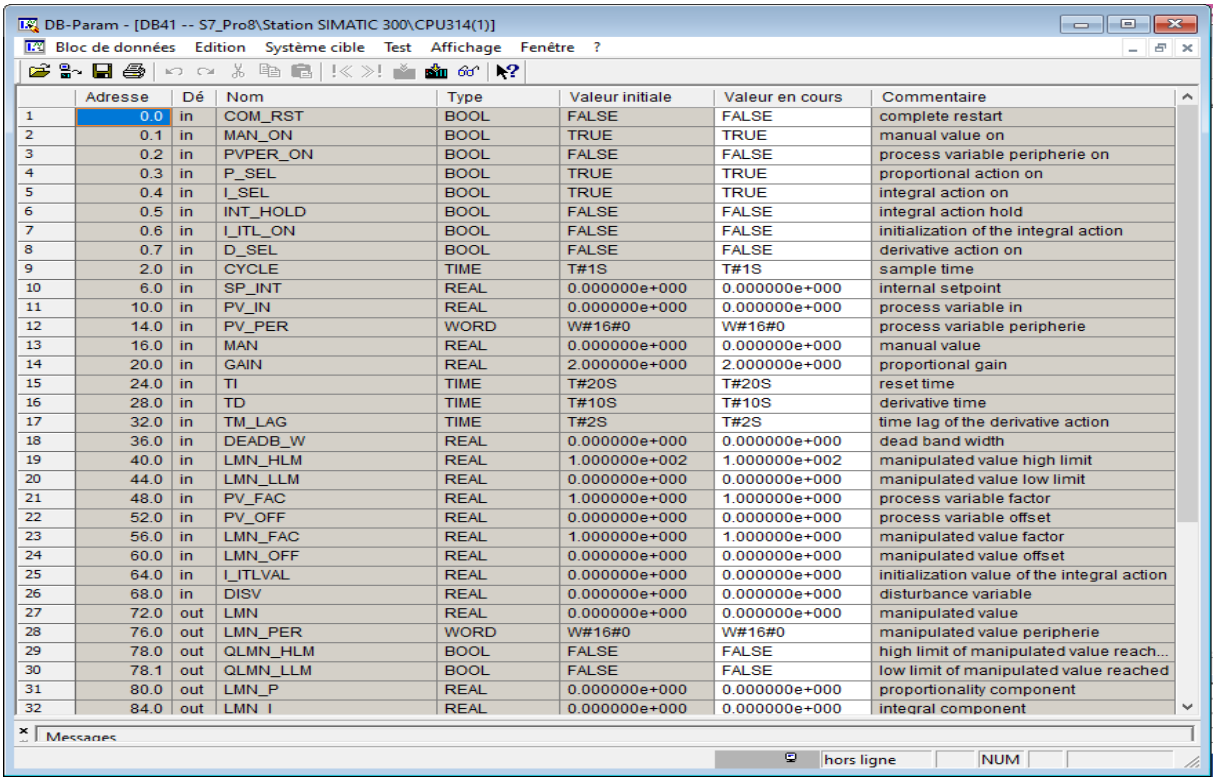


Figure 4.12 : Création d'un bloc de donnée



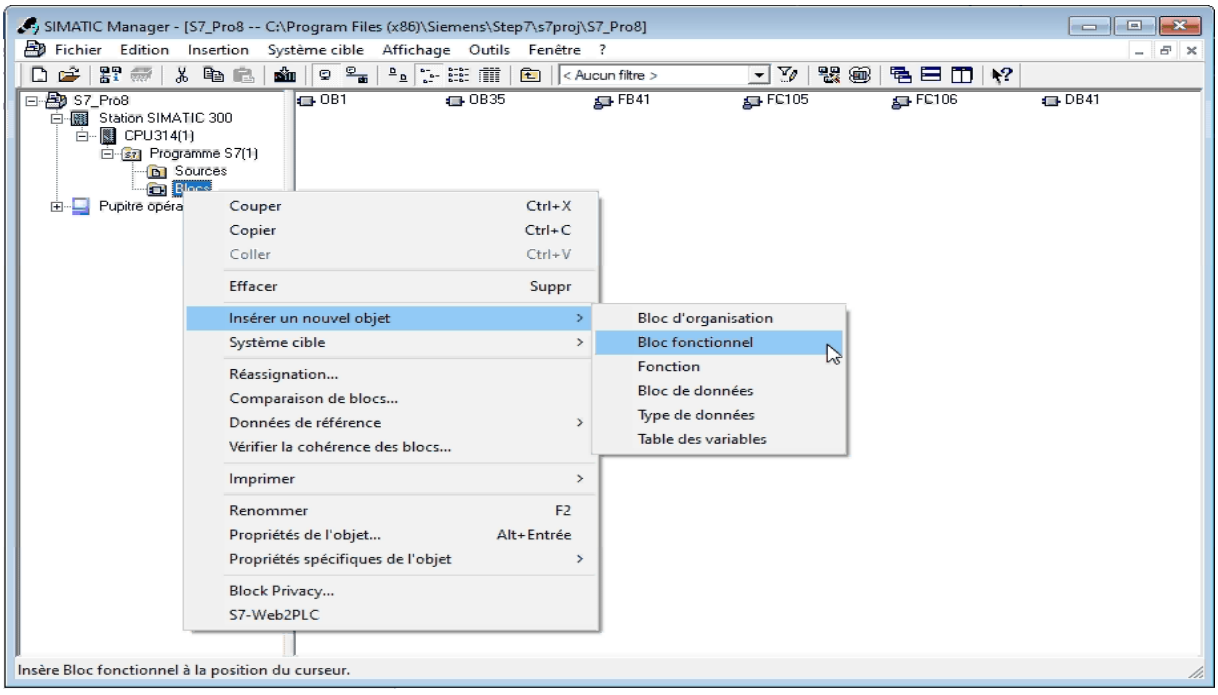
	Adresse	Dé	Nom	Type	Valeur initiale	Valeur en cours	Commentaire
1	0.0	in	COM_RST	BOOL	FALSE	FALSE	complete restart
2	0.1	in	MAN_ON	BOOL	TRUE	TRUE	manual value on
3	0.2	in	PVPER_ON	BOOL	FALSE	FALSE	process variable peripherie on
4	0.3	in	P_SEL	BOOL	TRUE	TRUE	proportional action on
5	0.4	in	I_SEL	BOOL	TRUE	TRUE	integral action on
6	0.5	in	INT_HOLD	BOOL	FALSE	FALSE	integral action hold
7	0.6	in	I_ITL_ON	BOOL	FALSE	FALSE	initialization of the integral action
8	0.7	in	D_SEL	BOOL	FALSE	FALSE	derivative action on
9	2.0	in	CYCLE	TIME	T#1S	T#1S	sample time
10	6.0	in	SP_INT	REAL	0.000000e+000	0.000000e+000	internal setpoint
11	10.0	in	PV_IN	REAL	0.000000e+000	0.000000e+000	process variable in
12	14.0	in	PV_PER	WORD	W#16#0	W#16#0	process variable peripherie
13	16.0	in	MAN	REAL	0.000000e+000	0.000000e+000	manual value
14	20.0	in	GAIN	REAL	2.000000e+000	2.000000e+000	proportional gain
15	24.0	in	TI	TIME	T#20S	T#20S	reset time
16	28.0	in	TD	TIME	T#10S	T#10S	derivative time
17	32.0	in	TM_LAG	TIME	T#2S	T#2S	time lag of the derivative action
18	36.0	in	DEADB_W	REAL	0.000000e+000	0.000000e+000	dead band width
19	40.0	in	LMN_HLM	REAL	1.000000e+002	1.000000e+002	manipulated value high limit
20	44.0	in	LMN_LLM	REAL	0.000000e+000	0.000000e+000	manipulated value low limit
21	48.0	in	PV_FAC	REAL	1.000000e+000	1.000000e+000	process variable factor
22	52.0	in	PV_OFF	REAL	0.000000e+000	0.000000e+000	process variable offset
23	56.0	in	LMN_FAC	REAL	1.000000e+000	1.000000e+000	manipulated value factor
24	60.0	in	LMN_OFF	REAL	0.000000e+000	0.000000e+000	manipulated value offset
25	64.0	in	I_ITLVAL	REAL	0.000000e+000	0.000000e+000	initialization value of the integral action
26	68.0	in	DISV	REAL	0.000000e+000	0.000000e+000	disturbance variable
27	72.0	out	LMN	REAL	0.000000e+000	0.000000e+000	manipulated value
28	76.0	out	LMN_PER	WORD	W#16#0	W#16#0	manipulated value peripherie
29	78.0	out	QLMN_HLM	BOOL	FALSE	FALSE	high limit of manipulated value reach...
30	78.1	out	QLMN_LLM	BOOL	FALSE	FALSE	low limit of manipulated value reached
31	80.0	out	LMN_P	REAL	0.000000e+000	0.000000e+000	proportionality component
32	84.0	out	LMN_I	REAL	0.000000e+000	0.000000e+000	integral component

Figure 4.13 : Contenu d'un bloc de donnée

4.3.3 Bloc de fonction FB

Le FB est à disposition via un espace mémoire correspondant. Si un FB est appelé, il lui est attribué un bloc de données (DB). On peut accéder aux données de cette instance DB par des appels depuis le FB. Un FB peut être attribué à différents DB. D'autres FB et d'autres FC peuvent être appelés dans un bloc de fonction par des commandes d'appel de blocs.

Création d'un FB



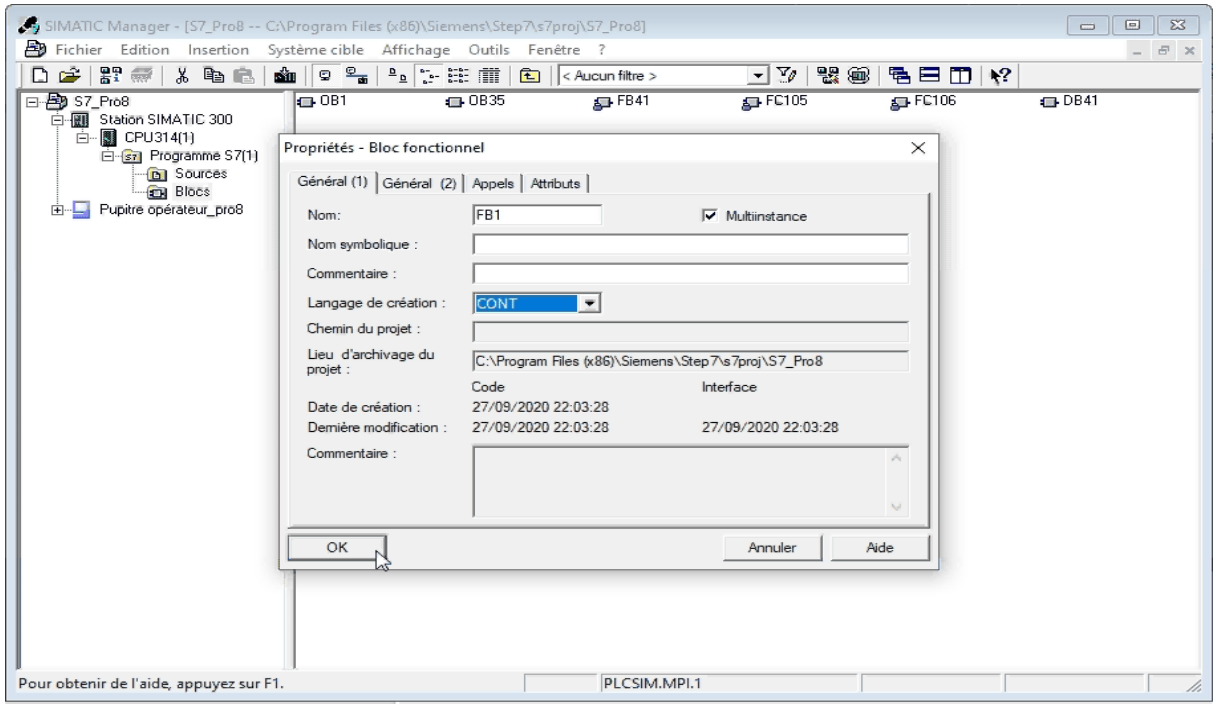
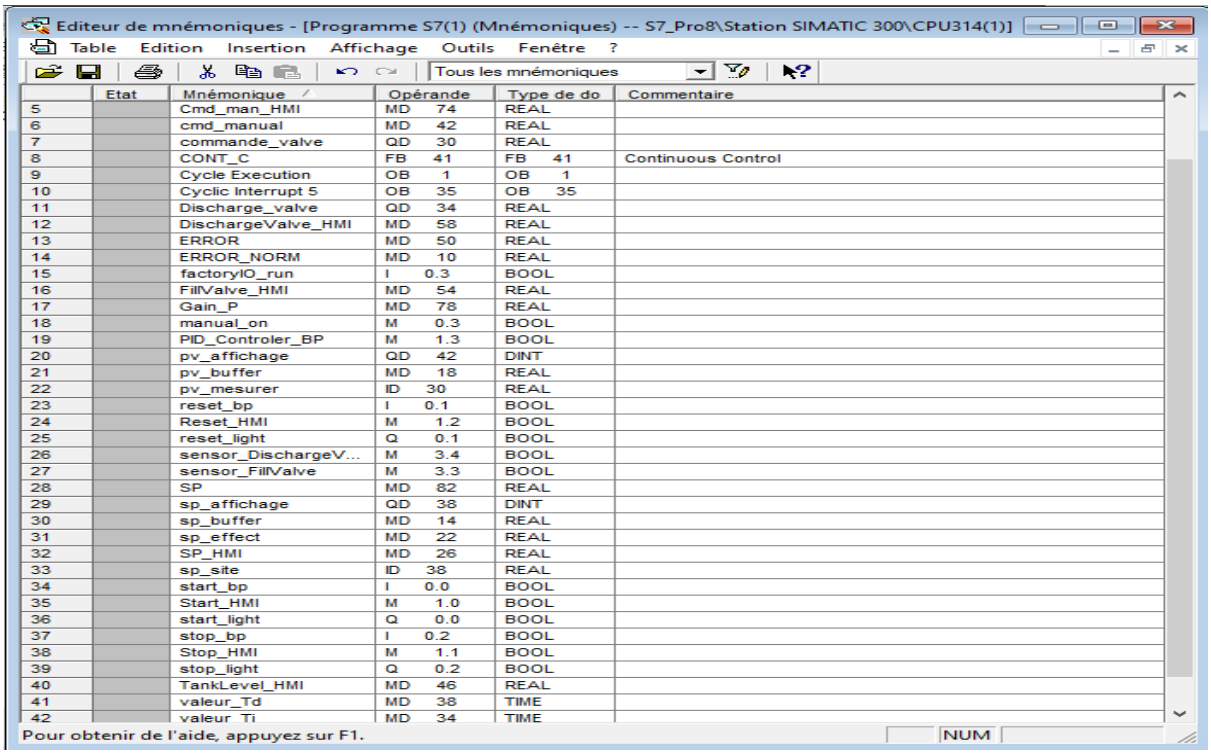


Figure 4.14 : Création d'un bloc de fonction

4.3.4 Table des mnémoniques

La table des mnémoniques nous permet de définir la liste des variables qui seront utilisées lors de la programmation. Il nous permet de désigner l'ensemble des entrées et sorties de notre système, la Figure 4.14 représente notre table des mnémoniques



	Etat	Mnémonique	Opérande	Type de do	Commentaire
5		Cmd_man_HMI	MD 74	REAL	
6		cmd_manual	MD 42	REAL	
7		commande_valve	QD 30	REAL	
8		CONT_C	FB 41	FB 41	Continuous Control
9		Cycle Execution	OB 1	OB 1	
10		Cyclic Interrupt 5	OB 35	OB 35	
11		Discharge_valve	QD 34	REAL	
12		DischargeValve_HMI	MD 58	REAL	
13		ERROR	MD 50	REAL	
14		ERROR_NORM	MD 10	REAL	
15		factoryIO_run	I 0.3	BOOL	
16		FillValve_HMI	MD 54	REAL	
17		Gain_P	MD 78	REAL	
18		manual_on	M 0.3	BOOL	
19		PID_Controler_BP	M 1.3	BOOL	
20		pv_affichage	QD 42	DINT	
21		pv_buffer	MD 18	REAL	
22		pv_mesurer	ID 30	REAL	
23		reset_bp	I 0.1	BOOL	
24		Reset_HMI	M 1.2	BOOL	
25		reset_light	Q 0.1	BOOL	
26		sensor_DischargeV...	M 3.4	BOOL	
27		sensor_FillValve	M 3.3	BOOL	
28		SP	MD 82	REAL	
29		sp_affichage	QD 38	DINT	
30		sp_buffer	MD 14	REAL	
31		sp_effect	MD 22	REAL	
32		SP_HMI	MD 26	REAL	
33		sp_site	ID 38	REAL	
34		start_bp	I 0.0	BOOL	
35		Start_HMI	M 1.0	BOOL	
36		start_light	Q 0.0	BOOL	
37		stop_bp	I 0.2	BOOL	
38		Stop_HMI	M 1.1	BOOL	
39		stop_light	Q 0.2	BOOL	
40		TankLevel_HMI	MD 46	REAL	
41		valeur_Td	MD 38	TIME	
42		valeur_Ti	MD 34	TIME	

Figure 4.15 : Table des mnémoniques

En programmant sur step7, nous travaillons avec des opérandes tels qu'E/S, mémentos, compteurs, temporisations, bloc de données, fonctions. Nous pouvons les adresser de manière absolue dans le programme, mais nous pouvons aussi améliorer considérablement la lisibilité et la clarté d'un programme en utilisant des mnémoniques à la place des adresses absolues.

Le cahier des charges :

L'élaboration du programme doit être respecte le cahier de charge suivant :

- Le départ, l'arrêt et le reset du processus ce fait à partir du IHM ou bien sur site. même aussi pour la consigne (SP).
- La détermination des paramètres de régulateur PID se fait seulement à partir de l'IHM.
- Visualisé les changements des variables SP et PV sur site.
- A partir de IHM on peut visualiser l'état d'avancement du processus (vanne de charge/décharge la consigne SP, la sortie PV, erreur, mode de fonctionnement).
- Déterminer le mode de fonctionnement (auto/manuel) à partir de l'IHM.
- Traçage des courbes des variables SP et PV en temps réel.
- L'archivage des variables SP et PV.

Condition d'activation et désactivation du premier réseau Ladder

$$CAX_1 = CAX_2 = I_{0.3}.(I_{0.0} + M_{1.0})$$
$$CDX_1 = CDX_2 = I_{0.3}.(I_{0.2} + M_{1.1})$$
$$CAX_3 = CAX_4 = I_{0.3}.((X_1 + X_2) + (I_{0.2} + M_{1.1}))$$
$$CDX_3 = CDX_4 = I_{0.3}.(I_{0.0} + M_{1.0})$$
$$CAX_5 = CAX_6 = I_{0.3}.((M_{1.1} + M_{1.2}) + (I_{0.1} + Q_{0.2}))$$
$$CDX_5 = CDX_6 = I_{0.3}.(I_{0.0} + M_{1.0})$$

$X_1 : start\ light$
$X_2 : start\ HMI$
$X_3 : stop\ light$
$X_4 : stop\ HMI$
$X_5 : reset\ light$
$X_6 : reset\ HMI$

4.3.5 Programmation avec Ladder

Le contenu du bloc OB1

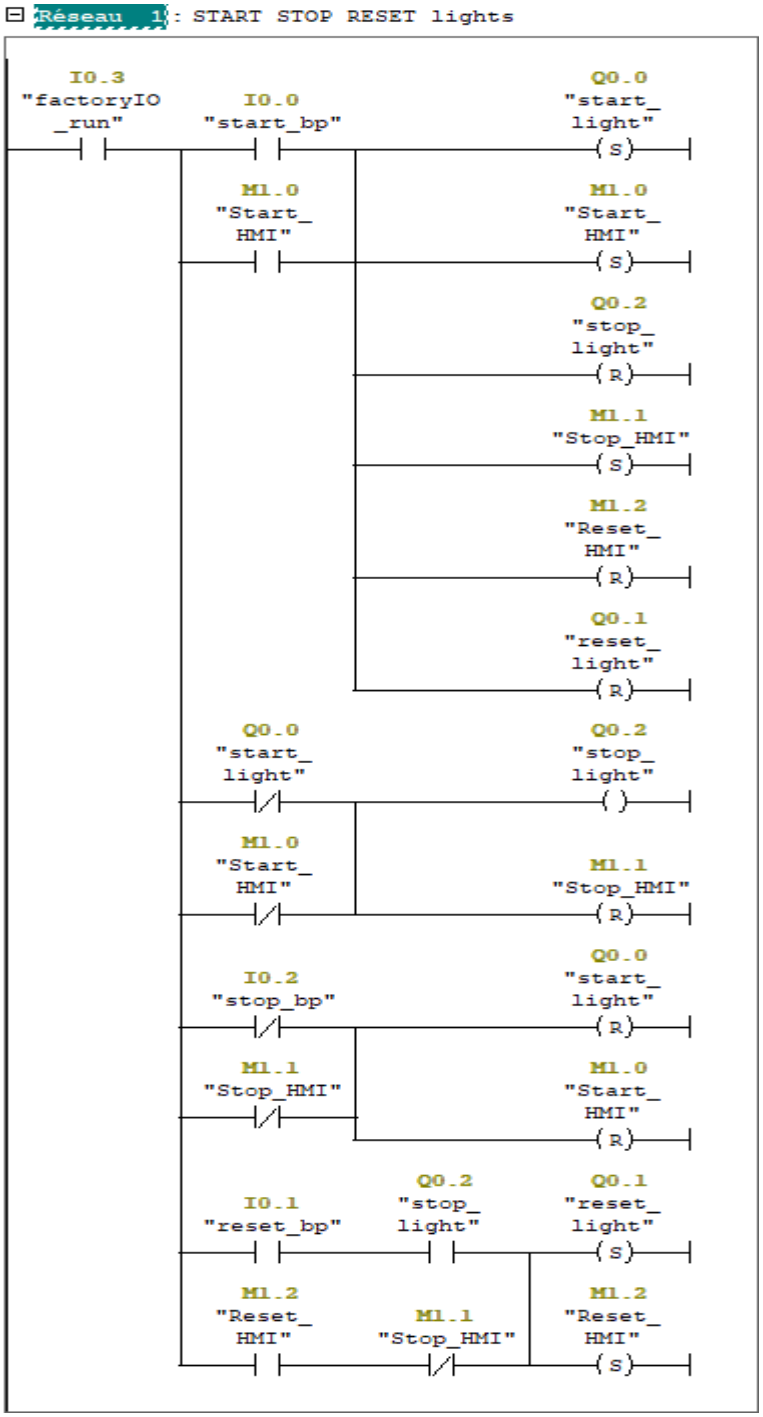


Figure 4.16 : Présentation de fonctionnement des lampes/bottons Start Stop et Reset

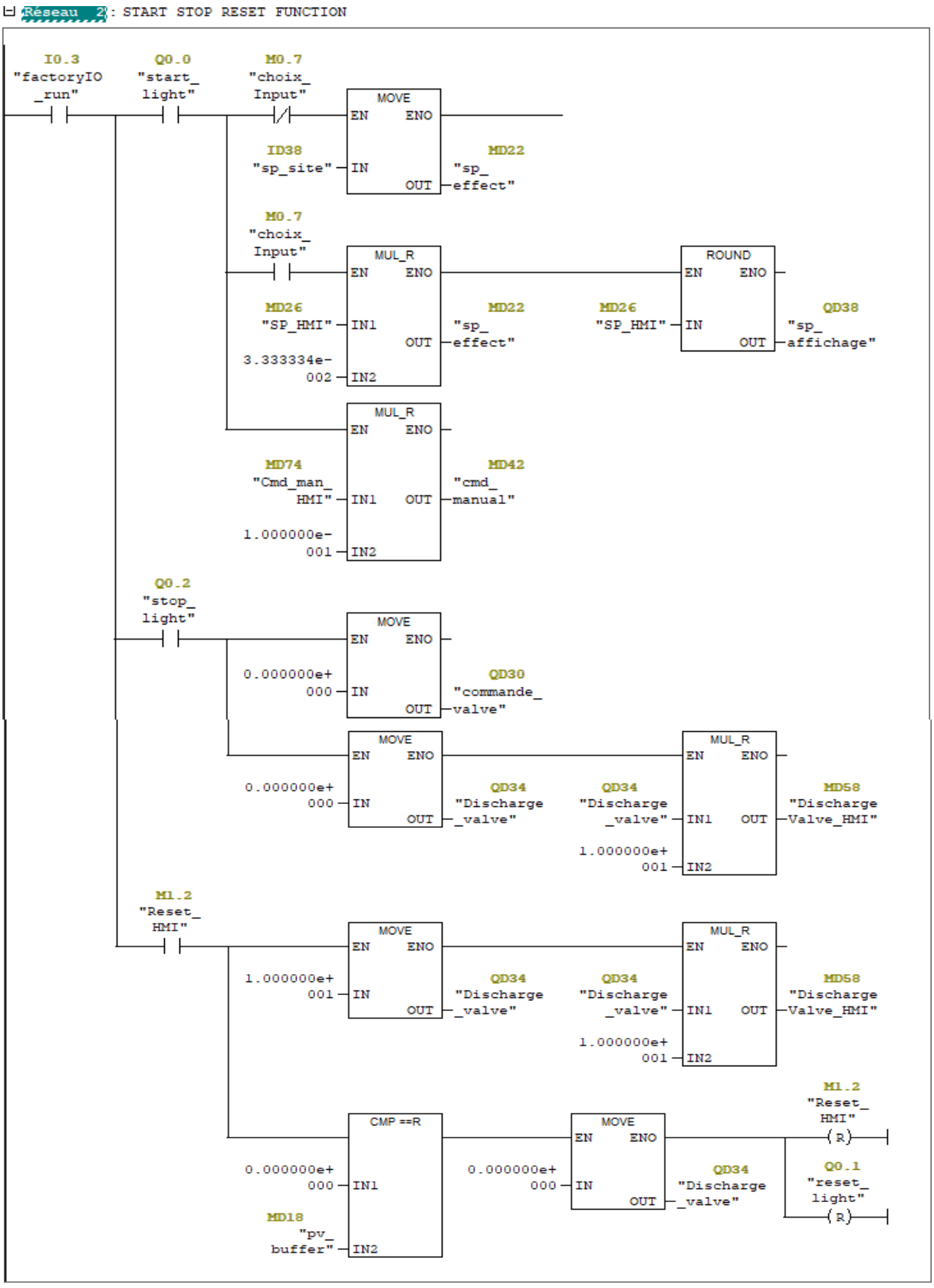


Figure 4.17 : Présentation des actions des bottons Start Stop et Reset

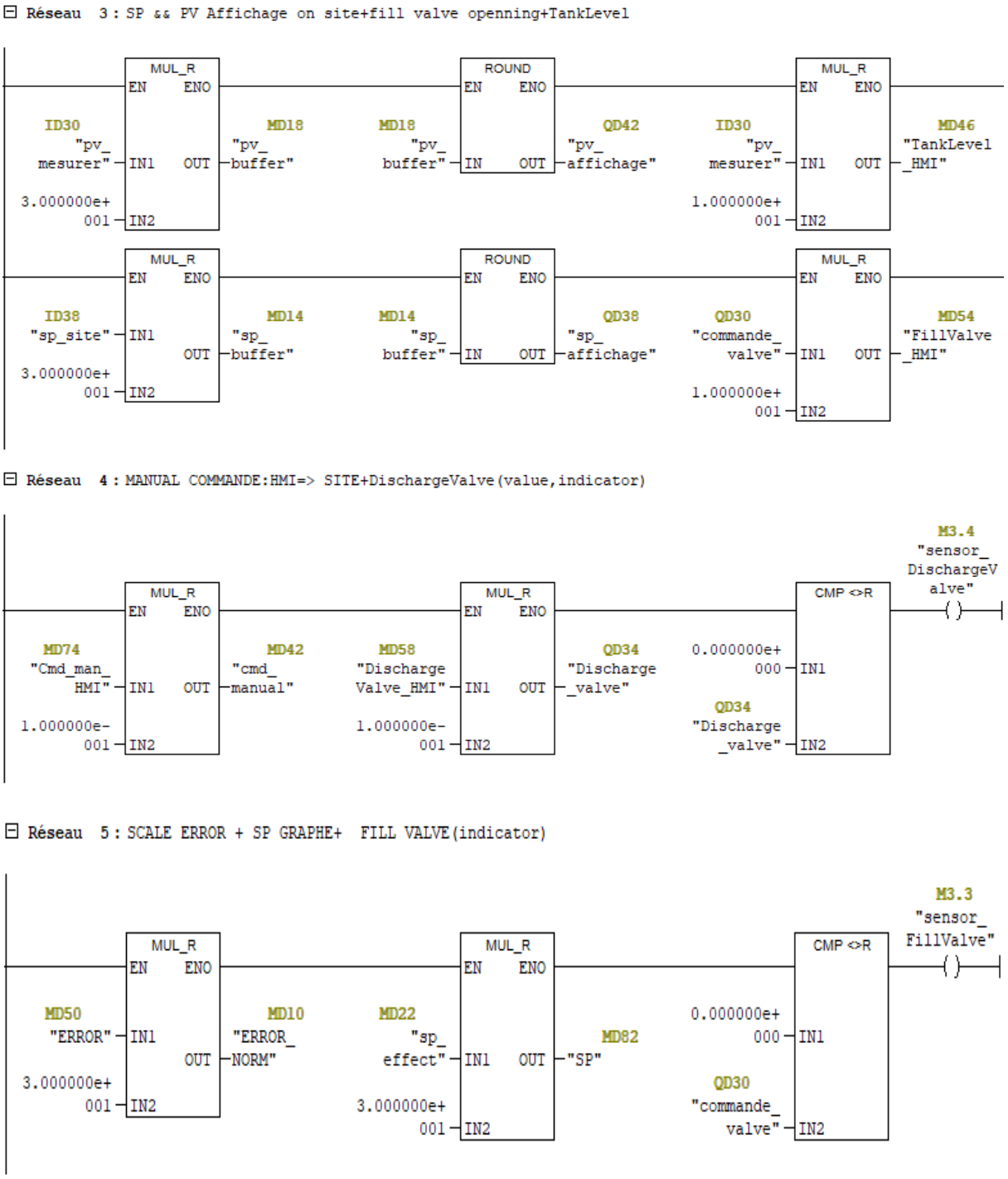


Figure 4.18 : Présentation des parties complémentaire du programme

Contenus du bloc OB35

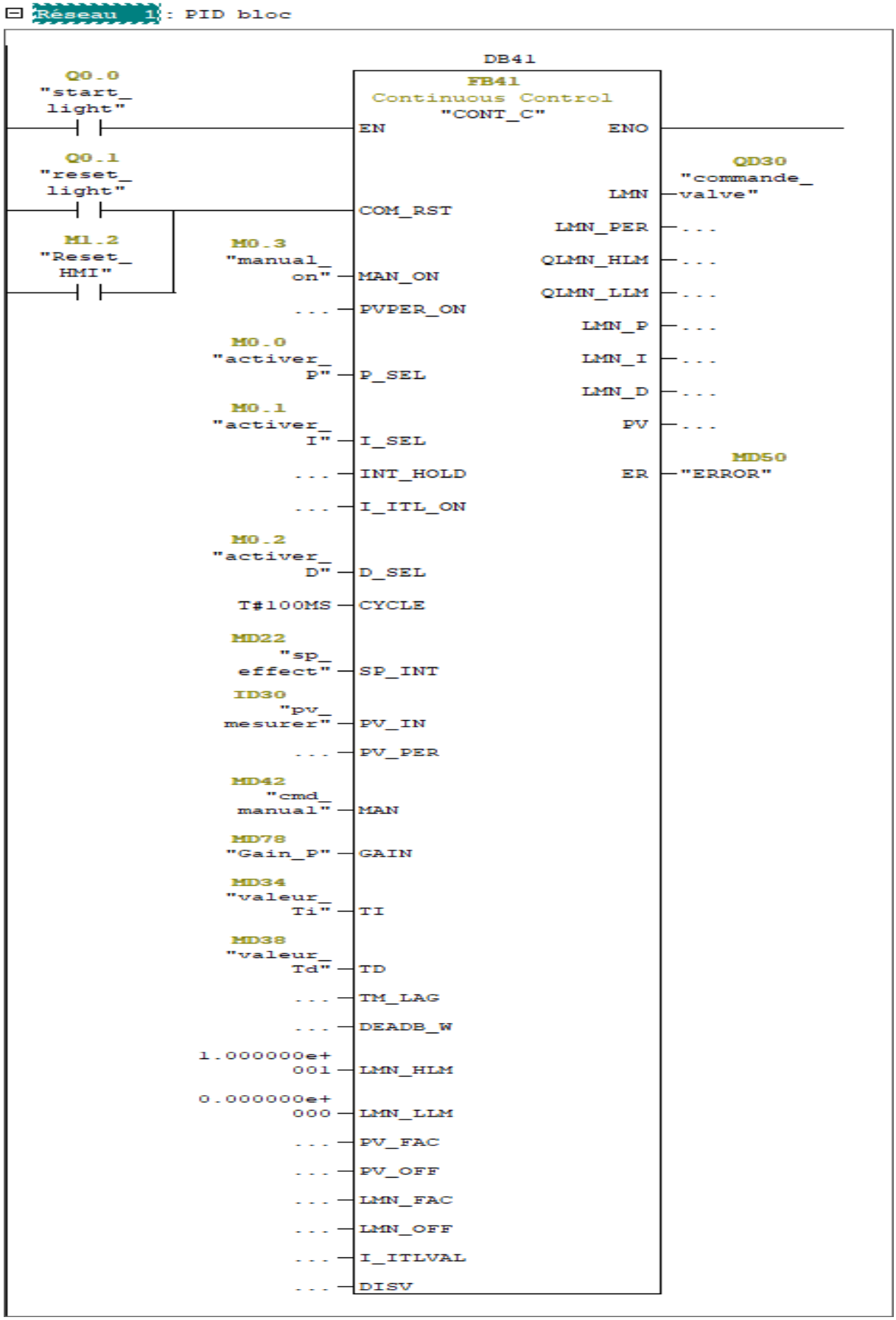


Figure 4.19 : le bloc PID - FB 41 « CONT_C »-

4.3.6 Programmation avec SCL

Contenus du bloc OB1

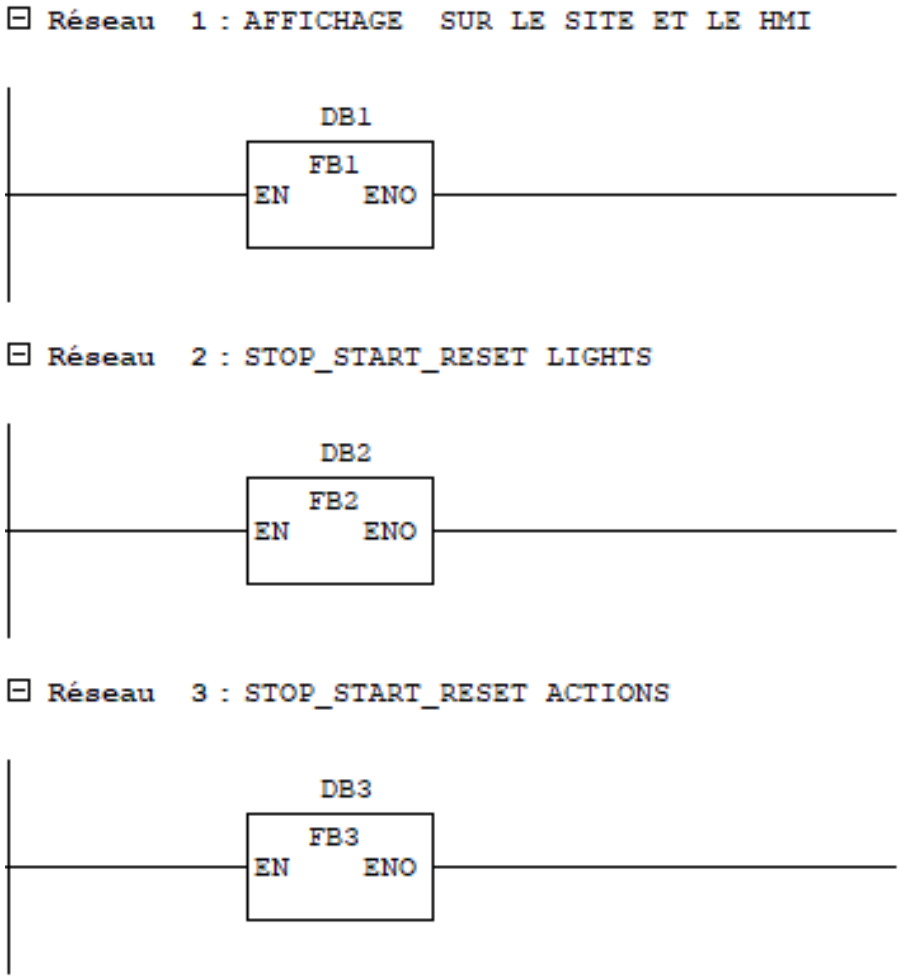


Figure 4.20 : Contenus du bloc OB1

Contenus du bloc OB35

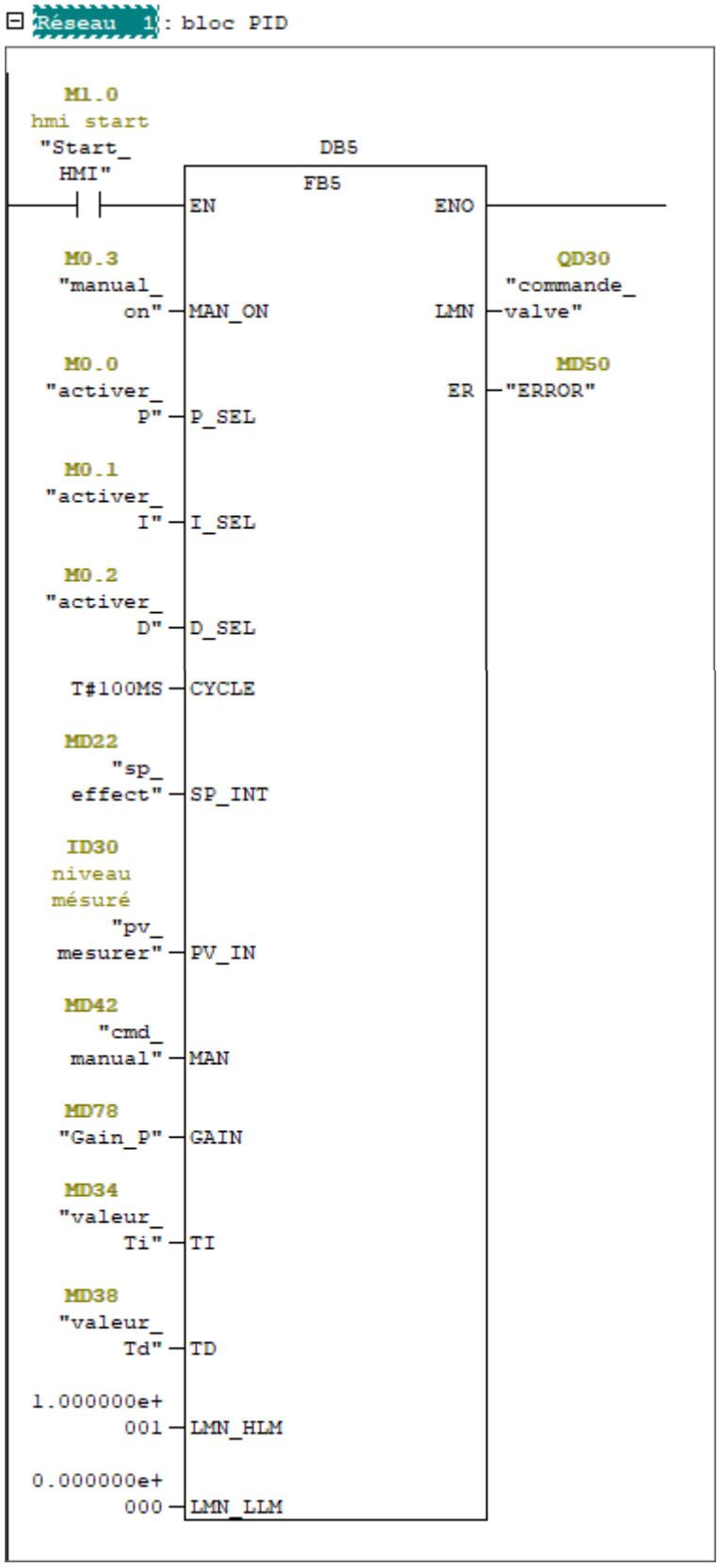


Figure 4.21 : Le bloc PID programmé avec SCL placé dans l'OB35

Contenus des blocs fonctionnels (FB1, FB2, FB3, FB5)

```
FUNCTION_BLOCK FB1

BEGIN
//////////////////////////////////////////
MD14:=30.0*ID38;            //SP AFFICHAGE
QD38:=REAL_TO_DINT (MD14); //**************
MD18:=30.0*ID30;            //PV AFFICHQGE
QD42:=REAL_TO_DINT (MD18); //**************
//////////////////////////////////////////
MD54:=10.0*QD30;          //FILLE VALVE HMI
MD46:=10.0*ID30;          //TANK LEVEL HMI
//////////////////////////////////////////
MD42:=0.1*MD74;       //COMMANDE MANUAL HMI
QD34:=0.1*MD58;       //DISCHARGE VALVE HMI
//////////////////////////////////////////
MD10:=30.0*MD50;     //ERROR
MD82:=30.0*MD22;    //SP GRAPHE
//////////////////////////////////////////
 IF QD34 <> 0.0 THEN  //************************
    M3.4 := TRUE;     //DISCHARGE VALVE SENSOR
 ELSE                 // **********************
    M3.4 := FALSE;    // **********************
 END_IF;              // **********************
//////////////////////////////////////////
 IF QD30 <> 0.0 THEN  //************************
    M3.3 := TRUE;     //************************
 ELSE                 //*FILL VALVE SENSOR ***
    M3.3 := FALSE;    //************************
 END_IF;              //************************
//////////////////////////////////////////
END_FUNCTION_BLOCK
```

Figure 4.22 : Programme SCL de FB1

```
FUNCTION_BLOCK  FB2
BEGIN
M3.0:=NOT M3.0 AND M3.0 ;
M3.1:=NOT M3.1 OR M3.1;
//////////////////////////////
 IF I0.3 = TRUE THEN   //// FACTOY_IO_RUN

   IF  (I0.0 OR M1.0) THEN

       Q0.0:=TRUE;  //START LIGHT_ON
       M1.0:=TRUE;  //START HMI_ON
       Q0.2:=FALSE; //STOP LIGHT_OFF
       M1.1:=TRUE;  //STOP HMI_OFF
       Q0.1:=FALSE; //RESET LIGHT_OFF
       M1.2:=FALSE; //RESET HMI_OFF
   END_IF;
   IF (NOT Q0.0) OR (NOT M1.0) THEN

        Q0.2:=TRUE;  // STOP LIGHT_ON
        M1.1:=FALSE; //STOP LIGHT HMI_ON
   END_IF;

   IF (NOT I0.2) OR (NOT M1.1) THEN

        Q0.0:=FALSE;  //START LIGHT_OFF
        M1.0:=FALSE;  //START HMI_OFF
   END_IF;
   IF (I0.1 AND Q0.2)OR (M1.2 AND NOT M1.1)  THEN

        Q0.1:=TRUE;  //RESET LIGHT_ON
        M1.2:=TRUE;  //RESET HMI_ON

   END_IF;

 END_IF;

END_FUNCTION_BLOCK
```

Figure 4.23 : Programme SCL de FB2

```
FUNCTION_BLOCK FB3
BEGIN
 IF I0.3 = TRUE THEN  //*** FACTORY IO RUN ******/////////////////
 ////////////////////////////////////////////// START BUTTON /////
   IF  Q0.0 = TRUE THEN
    MD42:=0.1*MD74;     //COMMANDE MANUAL
      IF M0.7 = TRUE THEN
        MD22 := 3.333334e-002*MD26;//SP EFFECTIVE FROM HMI
      ELSE
        MD22 := ID38;     //SP EFFECTIVE FROM SITE
      END_IF;

   END_IF;
////////////////////////////////////////////// STOP BUTTON /////
   IF Q0.2 = TRUE THEN

      QD30 := 0.0;        //COMMANDE VALVE
      QD34 := 0.0;        //DISCHARGE VALVE
      MD58 := 10.0*QD34; //DISCHARGE VALVE HMI

   END_IF;
////////////////////////////////////////////// RESET BUTTON /////
   IF M1.2 = TRUE THEN

      QD34 := 10.0;        //DISCHARGE VALVE
      MD58 := 10.0*QD34; //DISCHARGE VALVE HMI
       IF  MD18 = 0.0 THEN

        QD34 := 0.0;      //DISCHARGE VALVE MODE RESET
        M1.2 := FALSE;    //RESET HMI OFF
        Q0.1 := FALSE;    //RESET LIGHT OFF

       END_IF;
   END_IF;
 END_IF;////**************************************************///////

END_FUNCTION_BLOCK
```

Figure 4.24 : Programme SCL de FB3

```
FUNCTION_BLOCK FB5

VAR_INPUT
    MAN_ON,P_SEL,I_SEL,D_SEL:BOOL;
    CYCLE:TIME;
    SP_INT,PV_IN,MAN,GAIN:REAL;
    TI,TD:TIME;
    LMN_HLM,LMN_LLM:REAL;
END_VAR
VAR_OUTPUT
   LMN,ER:REAL;
END_VAR

VAR
   Up,Ui,Ud,KI,Kd,ERR_1:REAL;

END_VAR

BEGIN

ER := SP_INT-PV_IN;// ERROR(K)
////////////////////////////////////////////////
  IF MAN_ON = FALSE THEN
////////////////////////////////////////////////
//////////////////////////////////////ACTION P ///
      IF P_SEL = TRUE  THEN
         Up := GAIN*ER;
      ELSE
         Up := 0.0;
      END_IF;
////////////////////////////////////////////////
//////////////////////////////////////ACTION I ///
      IF I_SEL = TRUE THEN
        IF ER < 0.4 THEN
         KI := (GAIN*DINT_TO_REAL(TIME_TO_DINT(CYCLE)))/(2.0*DINT_TO_REAL(TIME_TO_DINT(TI)));
         Ui := KI*ER + KI*ERR_1 + Ui;
        END_IF;
      ELSE
         Ui := 0.0;
      END_IF;
////////////////////////////////////////////////
//////////////////////////////////////ACTION D ///
      IF D_SEL = TRUE THEN
        IF ER < 0.4 THEN
         Kd := (2.0*GAIN*DINT_TO_REAL(TIME_TO_DINT(TD)))/ DINT_TO_REAL(TIME_TO_DINT(CYCLE));
         Ud := Kd*ER-Kd*ERR_1-Ud;
        END_IF;
      ELSE
         Ud := 0.0;
      END_IF;
////////////////////////////////////////////////
    LMN := Up+Ui+Ud;        /// PID OUTPUT ///
////////////////////////////////////////////////
  ELSE
    LMN := MAN; //// MANUAL COMMANDE
  END_IF;
  ERR_1:= ER;

    IF (LMN < LMN_LLM) THEN         ///////////////////
        LMN := LMN_LLM;             //LIMITATION OF //
    ELSIF (LMN > LMN_HLM) THEN      ///  LMN VALUE ///
       LMN := LMN_HLM;              // [0V - 10V] ///
    END_IF;                        ///////////////////

END_FUNCTION_BLOCK
```

Figure 4.25 : Programme SCL de FB5

4.3.7 Programmation avec List

Contenus du bloc OB1

```
Réseau 1 : STOP START RESET LIGHTS

      A     "factoryIO_run"                  I0.3
      A(
      O     "start_bp"                       I0.0
      O     "Start_HMI"                      M1.0
      )
      S     "start_light"                    Q0.0
      S     "Start_HMI"                      M1.0
      R     "stop_light"                     Q0.2
      S     "Stop_HMI"                       M1.1
      R     "reset_light"                    Q0.1
      R     "Reset_HMI"                      M1.2
      A(
      ON    "start_light"                    Q0.0
      ON    "Start_HMI"                      M1.0
      )
      =     "stop_light"                     Q0.2
      R     "Stop_HMI"                       M1.1
      A(
      ON    "stop_bp"                        I0.2
      ON    "Stop_HMI"                       M1.1
      )
      R     "start_light"                    Q0.0
      R     "Start_HMI"                      M1.0
      A(
      A     "reset_bp"                       I0.1
      A     "stop_light"                     Q0.2
      O
      A     "Reset_HMI"                      M1.2
      AN    "Stop_HMI"                       M1.1
      )
      S     "reset_light"                    Q0.1
      S     "Reset_HMI"                      M1.2
```

Figure 4.26 : Présentation de fonctionnement des lampes/bottons Start Stop et Reset

```
Réseau 2 : START STOP RESET FONCTIONNEMENT

      A     "factoryIO_run"                  I0.3
      =     L     20.0
      A     L     20.0
      A     "start_light"                    Q0.0
      =     L     20.1
      A     L     20.1
      AN    "choix_Input"                    M0.7
      JNB   J1
      L     "sp_site"                        ID38
      T     "sp_effect"                      MD22
J1:   A(
      A     L     20.1
      A     "choix_Input"                    M0.7
      JNB   J2
      L     "SP_HMI"                         MD26
      L     3.333334e-002
      *R
      T     "sp_effect"                      MD22
      AN    OV
      SAVE
      CLR
```

```
J2:   A     BR
      )
      JNB   J3
      L     "SP_HMI"                    MD26
      RND
      T     "sp_affichage"              QD38
J3:   A     L     20.1
      JNB   J4
      L     "Cmd_man_HMI"               MD74
      L     1.000000e-001
      *R
      T     "cmd_manual"                MD42
J4:   A     L     20.0
      A     "stop_light"                Q0.2
      =     L     20.1
      A     L     20.1
      JNB   J5
      L     0.000000e+000
      T     "commande_valve"            QD30
J5:   A(
      A     L     20.1
      JNB   J6
      L     0.000000e+000
      T     "Discharge_valve"           QD34
      SET
      SAVE
      CLR
J6:   A     BR
      )
      JNB   J7
      L     "Discharge_valve"           QD34
      L     1.000000e+001
      *R
      T     "DischargeValve_HMI"        MD58
J7:   A     L     20.0
      A     "Reset_HMI"                 M1.2
      =     L     20.1
      A(
      A     L     20.1
      JNB   J8
      L     1.000000e+001
      T     "Discharge_valve"           QD34
      SET
      SAVE
      CLR
J8:   A     BR
      )
      JNB   J9
      L     "Discharge_valve"           QD34
      L     1.000000e+001
      *R
      T     "DischargeValve_HMI"        MD58
J9:   A     L     20.1
      A(
      L     0.000000e+000
      L     "pv_buffer"                 MD18
      ==R
      )
      JNB   J10
      L     0.000000e+000
      T     "Discharge_valve"           QD34
      SET
      SAVE
      CLR
J10:  A     BR
      R     "Reset_HMI"                 M1.2
      R     "reset_light"               Q0.1
```

Figure 4.27 : Présentation des actions des bottons Start Stop et Reset

```
Réseau 3 : AFFICHAGE COVERTION
      L     "sp_site"                          ID38
      L     3.000000e+001
      *R
      T     "sp_buffer"                        MD14

      L     "sp_buffer"                        MD14
      RND
      T     "sp_affichage"                     QD38

      L     "pv_mesurer"                       ID30
      L     3.000000e+001
      *R
      T     "pv_buffer"                        MD18

      L     "pv_buffer"                        MD18
      RND
      T     "pv_affichage"                     QD42

      L     "commande_valve"                   QD30
      L     1.000000e+001
      *R
      T     "FillValve_HMI"                    MD54

      L     "pv_mesurer"                       ID30
      L     1.000000e+001
      *R
      T     "TankLevel_HMI"                    MD46

      L     "Cmd_man_HMI"                      MD74
      L     1.000000e-001
      *R
      T     "cmd_manual"                       MD42

      L     "DischargeValve_HMI"               MD58
      L     1.000000e-001

      *R
      T     "Discharge_valve"                  QD34

      =     "sensor_DischargeValve"            M3.4

      L     0.000000e+000
      L     "commande_valve"                   QD30
      <>R
      =     "sensor_FillValve"                 M3.3

      L     "ERROR"                            MD50
      L     3.000000e+001
      *R
      T     "ERROR_NORM"                       MD10

      L     "sp_effect"                        MD22
      L     3.000000e+001
      *R
      T     "SP"                               MD82
```

Figure 4.28 : Présentation des parties complémentaires du programme

Contenus de bloc OB35

Réseau 1 : PID

```
      A(
      O     "reset_light"                    Q0.1
      O     "Reset_HMI"                      M1.2
      )
      =     L     20.0

      A     "manual_on"                      M0.3
      =     L     20.1

      A     "false"                          M3.0
      =     L     20.2

      A     "activer_P"                      M0.0
      =     L     20.3

      A     "activer_I"                      M0.1
      =     L     20.4

      A     "activer_D"                      M0.2
      =     L     20.7
      A     "start_light"                    Q0.0
      JNB   J10
      CALL  "CONT_C" , DB41                  FB41
       COM_RST :=L20.0
       MAN_ON  :=L20.1
       PVPER_ON:=L20.2
       P_SEL   :=L20.3
       I_SEL   :=L20.4
       INT_HOLD:=
       I_ITL_ON:=
       D_SEL   :=L20.7
       CYCLE   :=T#100MS
       SP_INT  :="sp_effect"                 MD22
       PV_IN   :="pv_mesurer"                ID30
       PV_PER  :=
       MAN     :="cmd_manual"                MD42
       GAIN    :="Gain_P"                    MD78
       TI      :="valeur_Ti"                 MD34
       TD      :="valeur_Td"                 MD38
       TM_LAG  :=
       DEADB_W :=
       LMN_HLM :=1.000000e+001
       LMN_LLM :=0.000000e+000
       PV_FAC  :=
       PV_OFF  :=
       LMN_FAC :=
       LMN_OFF :=
       I_ITLVAL:=
       DISV    :=
       LMN     :="commande_valve"            QD30
       LMN_PER :=
       QLMN_HLM:=
       QLMN_LLM:=
       LMN_P   :=
       LMN_I   :=
       LMN_D   :=
       PV      :=
       ER      :="ERROR"                     MD50
J10:  NOP   0
```

Figure 4.29 : Le bloc PID - FB 41 « CONT_C »-

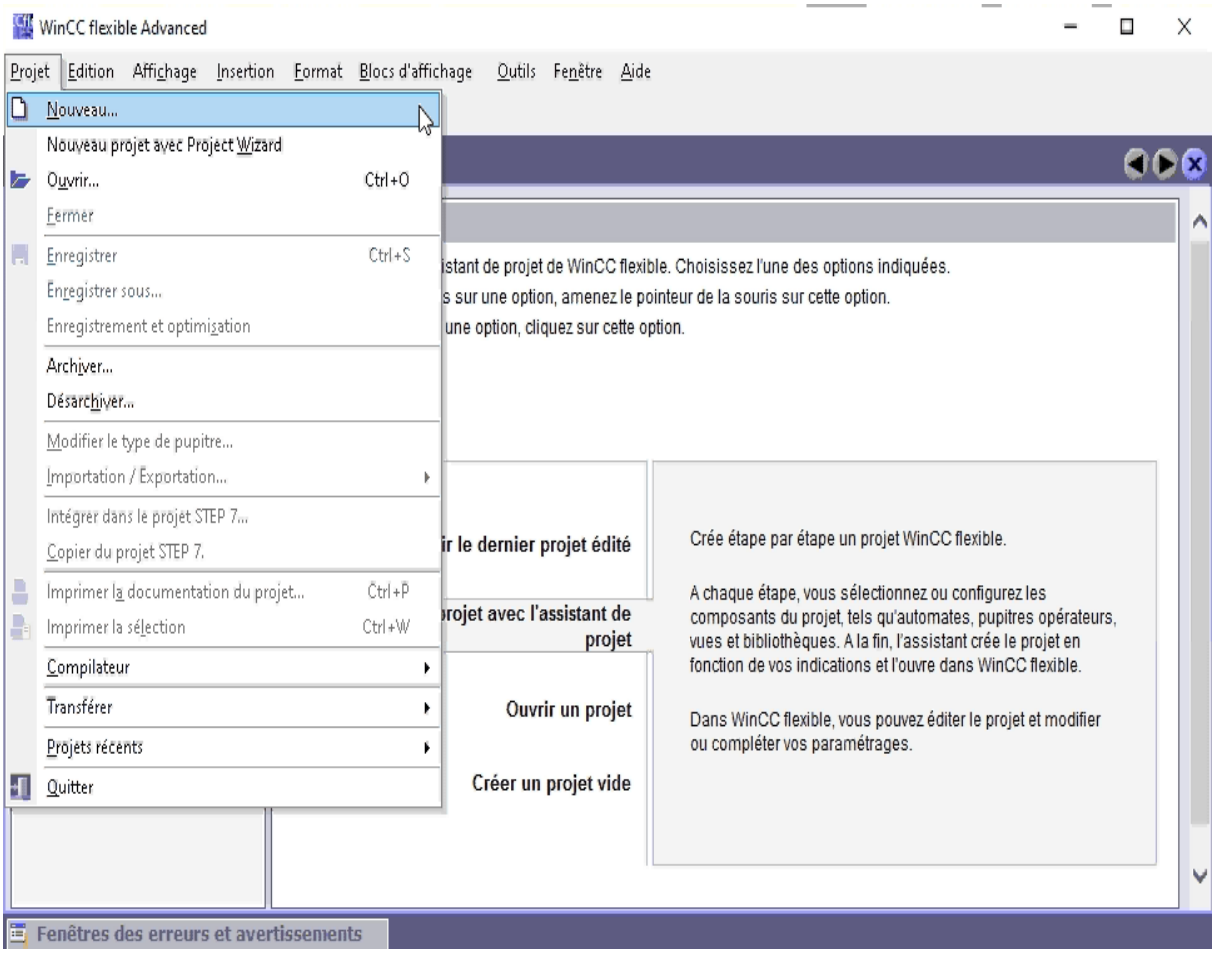
4.4 Présentation de la partie IHM

4.4.1 Création d'une station IHM

Une Interface Homme-Machine (IHM) est une interface utilisateur permettant de connecter une personne à une machine, à un système ou à un appareil. En théorie, il est donc possible d'utiliser ce terme pour définir n'importe quel écran permettant à un utilisateur d'interagir avec un appareil. Cependant, il est généralement utilisé pour le contexte d'un processus industriel.

Les IHM permettent principalement d'afficher des informations de façon visuelle pour permettre à l'utilisateur de superviser un processus industriel.

Les IHM peuvent prendre différentes formes. Il peut s'agir d'écrans directement intégrés aux machines, d'écrans d'ordinateur, de tablettes tactiles, et bien plus encore.



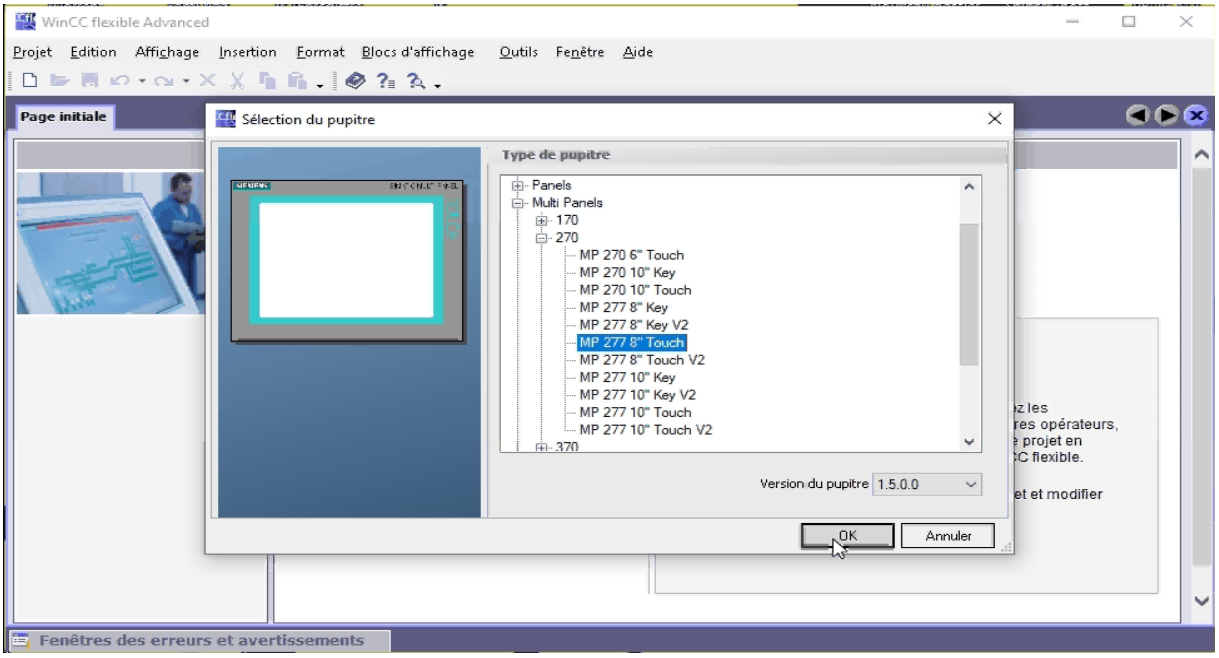


Figure 4.30 : Création d'une station IHM

4.4.2 Création de la table des variables

Dans la table des variables IHM On distingue deux types de variables, les variables externes et les variables internes :

- Les variables externes permettent de communiquer et d'échanger des données entre les composants d'un processus automatisé, entre un pupitre opérateur et un automate.
- Les variables internes ne possède aucun lien avec l'automate, elles sont enregistrer dans la mémoire du pupitre.

La Figure 4.30 est une représentation de notre table des variables IHM.

Variables

Nom	Connexion	Type de données	Mnémonique	Adresse	Cycle d'acqu...	Archive de don...	Mode d'acquisit...	Cycle d'arch...
activer_D	CPU314(1)	Bool	<indéfini>	M 0.2	100 ms	<indéfini>	Cyclique en continu	<indéfini>
activer_I	CPU314(1)	Bool	activer_I	M 0.1	100 ms	<indéfini>	Cyclique en continu	<indéfini>
activer_P	CPU314(1)	Bool	activer_P	M 0.0	100 ms	<indéfini>	Cyclique en continu	<indéfini>
choix_Input	CPU314(1)	Bool	choix_Input	M 0.7	100 ms	<indéfini>	Cyclique en continu	<indéfini>
Cmd_man_HMI	CPU314(1)	Real	Cmd_man_HMI	MD 74	100 ms	<indéfini>	Cyclique en continu	<indéfini>
cmd_manual	CPU314(1)	Real	cmd_manual	MD 42	100 ms	<indéfini>	Cyclique en continu	<indéfini>
DischargeValve_HMI	CPU314(1)	Real	DischargeValve_HMI	MD 58	100 ms	<indéfini>	Cyclique en continu	<indéfini>
ERROR	CPU314(1)	Real	ERROR	MD 50	100 ms	<indéfini>	Cyclique en continu	<indéfini>
ERROR_NORM	CPU314(1)	Real	ERROR_NORM	MD 10	100 ms	<indéfini>	Cyclique en continu	<indéfini>
FACEPLATE_PID	<Variable interne>	Bool	<indéfini>	<Pas d'adresse>	1 s	<indéfini>	Cyclique en continu	<indéfini>
FillValve_HMI	CPU314(1)	Real	FillValve_HMI	MD 54	100 ms	<indéfini>	Cyclique en continu	<indéfini>
Gain_P	CPU314(1)	Real	Gain_P	MD 78	100 ms	<indéfini>	Cyclique en continu	<indéfini>
manual_on	CPU314(1)	Bool	manual_on	M 0.3	100 ms	<indéfini>	Cyclique en continu	<indéfini>
PID_botton	<Variable interne>	Bool	<indéfini>	<Pas d'adresse>	1 s	<indéfini>	Cyclique en continu	<indéfini>
pv_affichage	CPU314(1)	DInt	pv_affichage	QD 42	100 ms	Archive_ladder_PV	Cyclique en continu	Cycle_1
Reset_HMI	CPU314(1)	Bool	Reset_HMI	M 1.2	100 ms	<indéfini>	Cyclique en continu	<indéfini>
sensor_DischargeValve	CPU314(1)	Bool	sensor_DischargeValve	M 3.4	100 ms	<indéfini>	Cyclique en continu	<indéfini>
sensor_FillValve	CPU314(1)	Bool	sensor_FillValve	M 3.3	100 ms	<indéfini>	Cyclique en continu	<indéfini>
SP	CPU314(1)	Real	SP	MD 82	100 ms	Archive_ladder_SP	Cyclique en continu	Cycle_1
sp_affichage	CPU314(1)	DInt	sp_affichage	QD 38	100 ms	<indéfini>	Cyclique en continu	<indéfini>
sp_buffer	CPU314(1)	Real	sp_buffer	MD 14	100 ms	<indéfini>	Cyclique en continu	<indéfini>
SP_HMI	CPU314(1)	Real	SP_HMI	MD 26	100 ms	<indéfini>	Cyclique en continu	<indéfini>
Start_HMI	CPU314(1)	Bool	Start_HMI	M 1.0	100 ms	<indéfini>	Cyclique en continu	<indéfini>
Stop_HMI	CPU314(1)	Bool	Stop_HMI	M 1.1	100 ms	<indéfini>	Cyclique en continu	<indéfini>
TankLevel_HMI	CPU314(1)	Real	TankLevel_HMI	MD 46	100 ms	<indéfini>	Cyclique en continu	<indéfini>
valeur_Td	CPU314(1)	Time	valeur_Td	MD 38	100 ms	<indéfini>	Cyclique en continu	<indéfini>
valeur_Ti	CPU314(1)	Time	valeur_Ti	MD 34	100 ms	<indéfini>	Cyclique en continu	<indéfini>

Figure 4.31 : Table des variables IHM

4.4.3 Archivage des variables

Elle permet d'archiver la consigne de niveau d'eau et la valeur du procès.

Pour faire l'archivage d'une variable, cliquez sur HISTORIQUE >>ARCHIVES créé un archive et dans la table des variable puis sélectionnée la variable a archivée via archive des donnée

Variables Archives

Nom	Nombre d'en...	Lieu d'archivage	Chemin	Méthode archivage	Nombr...	Niveau de ...	Activer l'ar...	Réponse au démarrage ...
Archive_ladder_PV	1000	Fichier - CSV (ASCII)	\Storage Card\Logs	Archive cyclique	10	90	Activée	Adjoindre à l'archive
Archive_ladder_SP	1000	Fichier - CSV (ASCII)	\Storage Card\	Archive cyclique	10	90	Activée	Adjoindre à l'archive

Variables Archives

VARIABLES

Nom	Connexion	Type de données	Mnémonique	Adresse	Cycle d'ac...	Archive de données	Mode d'acquisition a...	Cycle d'archivage
SP	CPU314(1)	Real	SP	MD 82	100 ms	Archive_ladder_SP	Cyclique en continu	Cycle_1
pv_affichage	CPU314(1)	DInt	pv_affichage	QD 42	100 ms	Archive_ladder_PV	Cyclique en continu	Cycle_1
PID_botton	<Variable interne>	Bool	<indéfini>	<Pas d'adre...	1 s	<indéfini>	Cyclique en continu	<indéfini>
sp_affichage	CPU314(1)	DInt	sp_affichage	QD 38	100 ms	<indéfini>	Cyclique en continu	<indéfini>
Stop_HMI	CPU314(1)	Bool	Stop_HMI	M 1.1	100 ms	<indéfini>	Cyclique en continu	<indéfini>
sensor_FillValve	CPU314(1)	Bool	sensor_FillValve	M 3.3	100 ms	<indéfini>	Cyclique en continu	<indéfini>

Figure 4.32 : Archivage des variables

4.4.4 La vue principale

C'est la vue qui s'affiche au lancement. Elle est principale et représente le système à réservoir et ses états. Depuis cette vue l'operateur peut accéder à la vue secondaire.

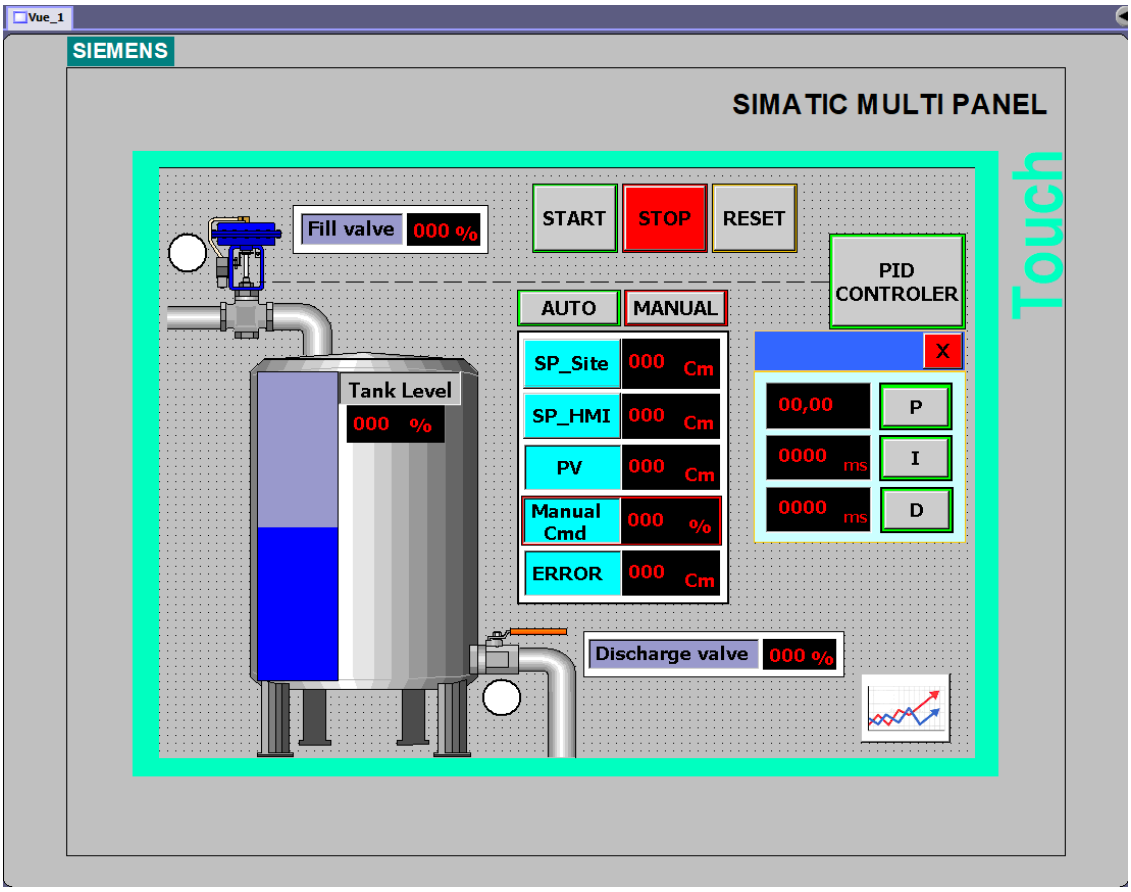


Figure 4.33 : La vue principale de l'IHM

4.4.5 La vue secondaire (les graphes)

C'est une vue détaillé elle représente les allures de la consigne « SP » et de la variable de mesure « PV ».

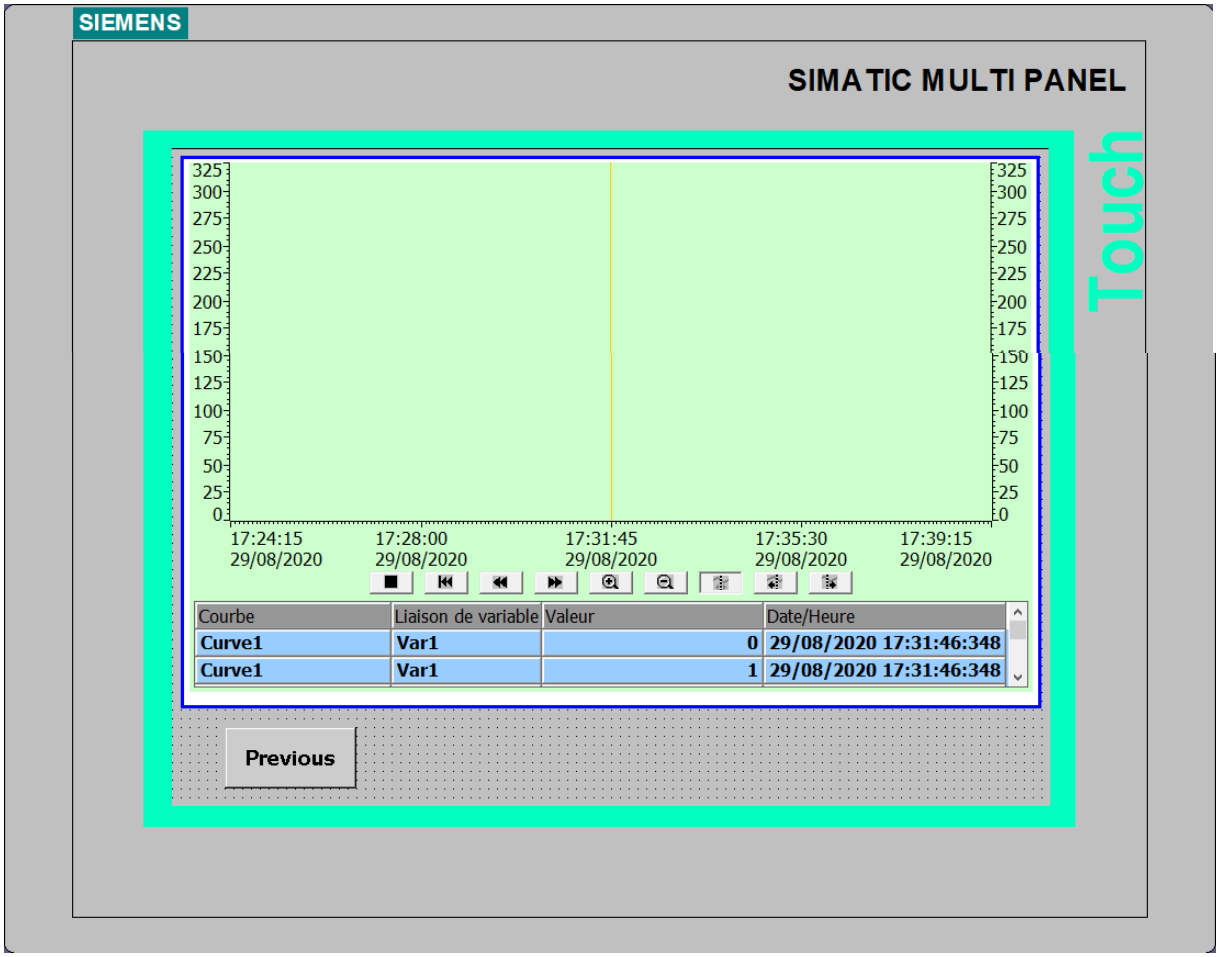


Figure 4.34 : La vue secondaire

4.5 Tests et simulations

4.5.1 Simulation du programme sous STEP7 avec PLCSIM

L'application de simulation S7-PLCSIM nous a permet d'exécuter et de tester notre programme qu'on a simulé sur ordinateur.

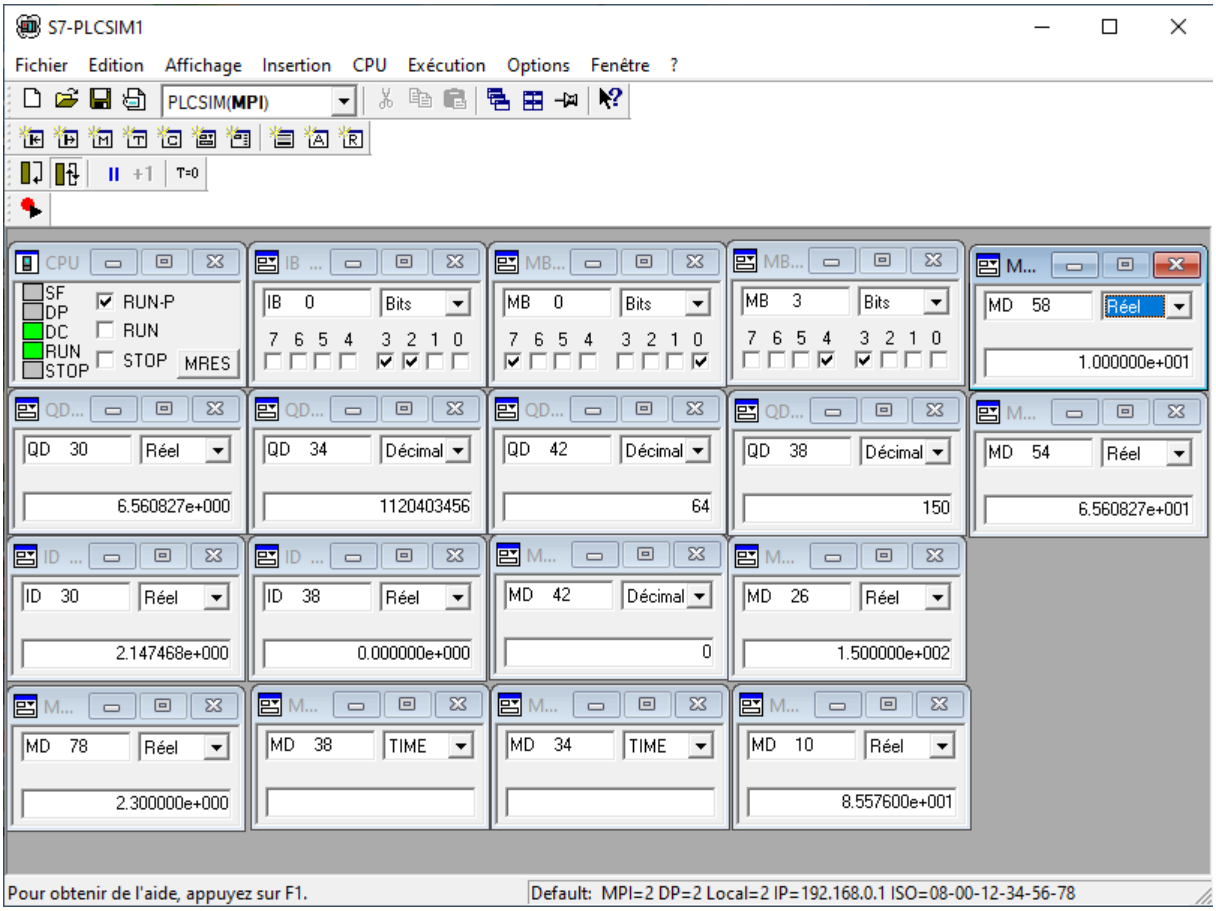


Figure 4.35 : Simulation avec le PLCSIM

4.5.2 Simulation du programme sous WinCC flexible

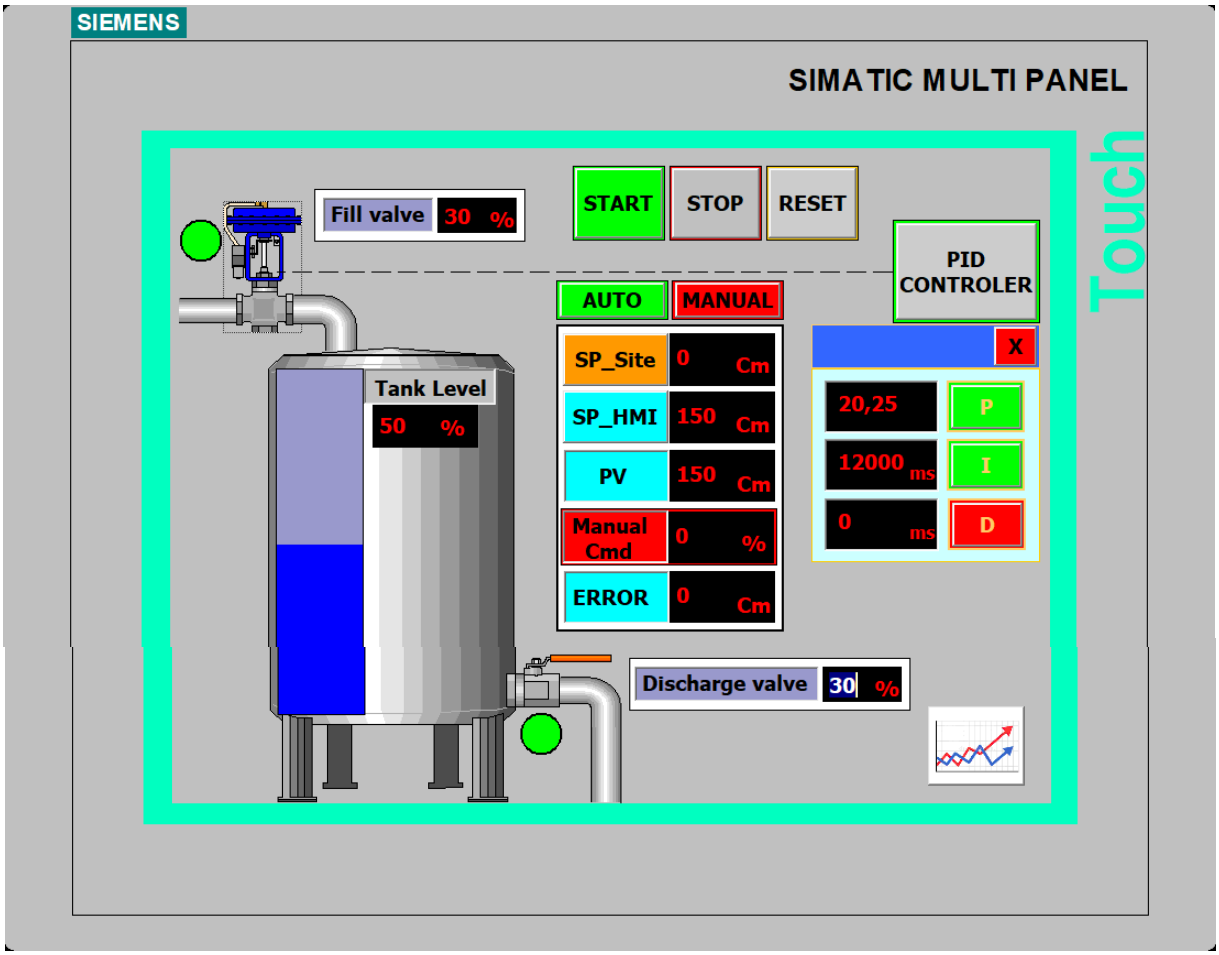


Figure 4.36 : Simulation avec WinCC flexible-vue générale

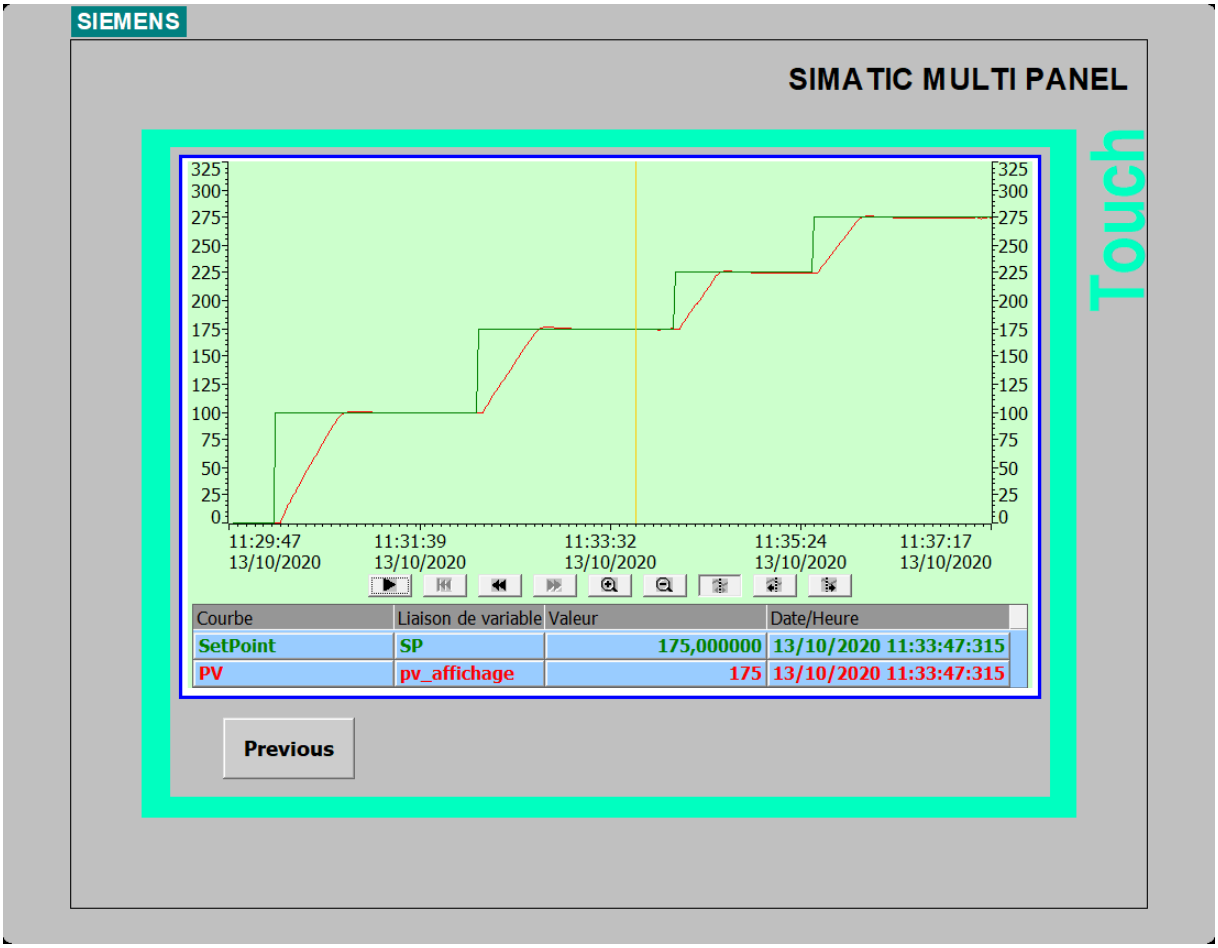


Figure 4.37 : Graphes de simulation avec WinCC flexible-vue secondaire

4.5.3 Réponse indicielle

- **Avec Ladder**

Dans la simulation nous avons utilisé ces valeurs Sp=150 cm, Kp=20,25 Ti=12s, vanne de décharge = 30%.

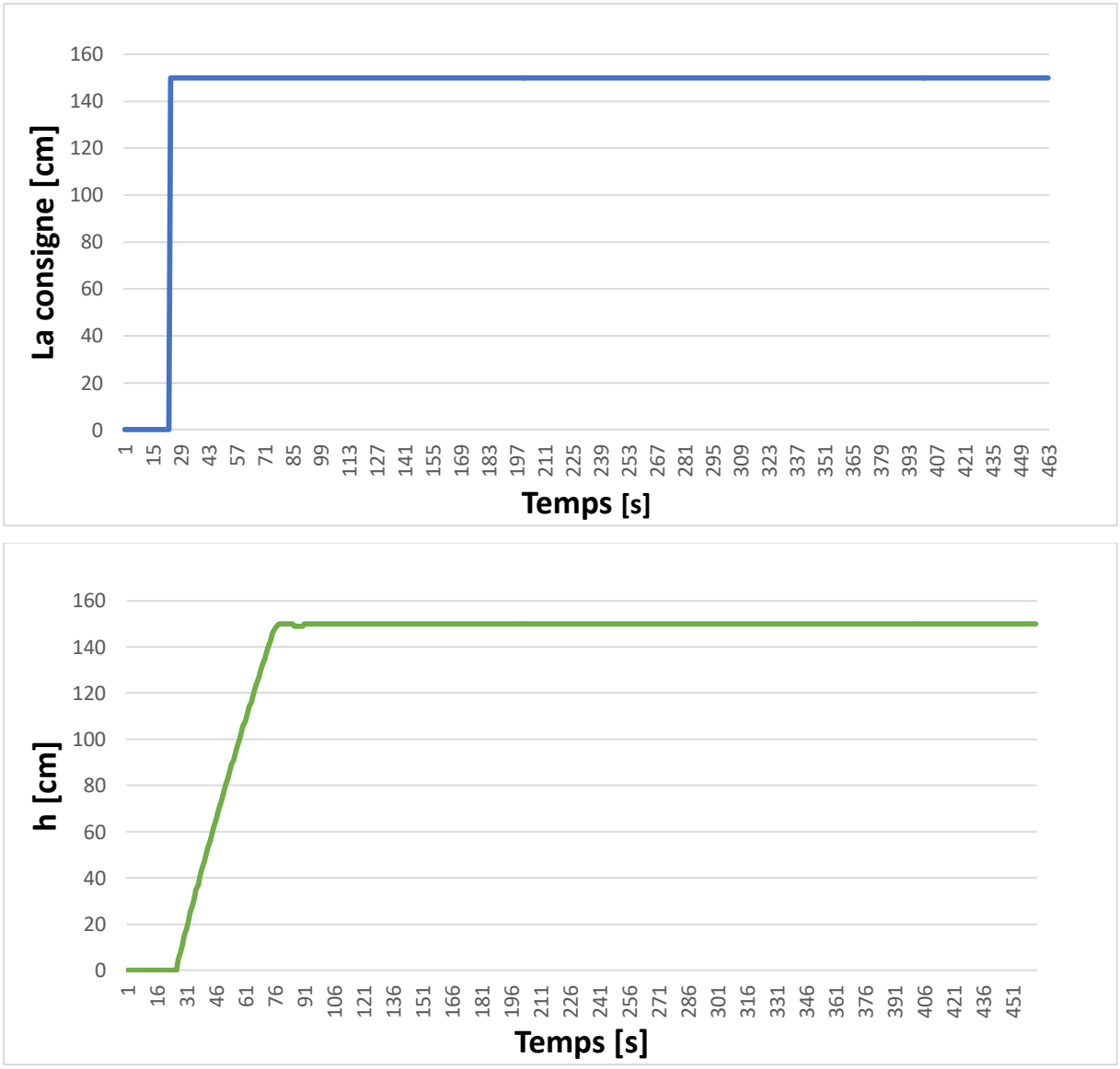


Figure 4.38 : résultats de simulation avec Ladder

➢ **Avec SCL**

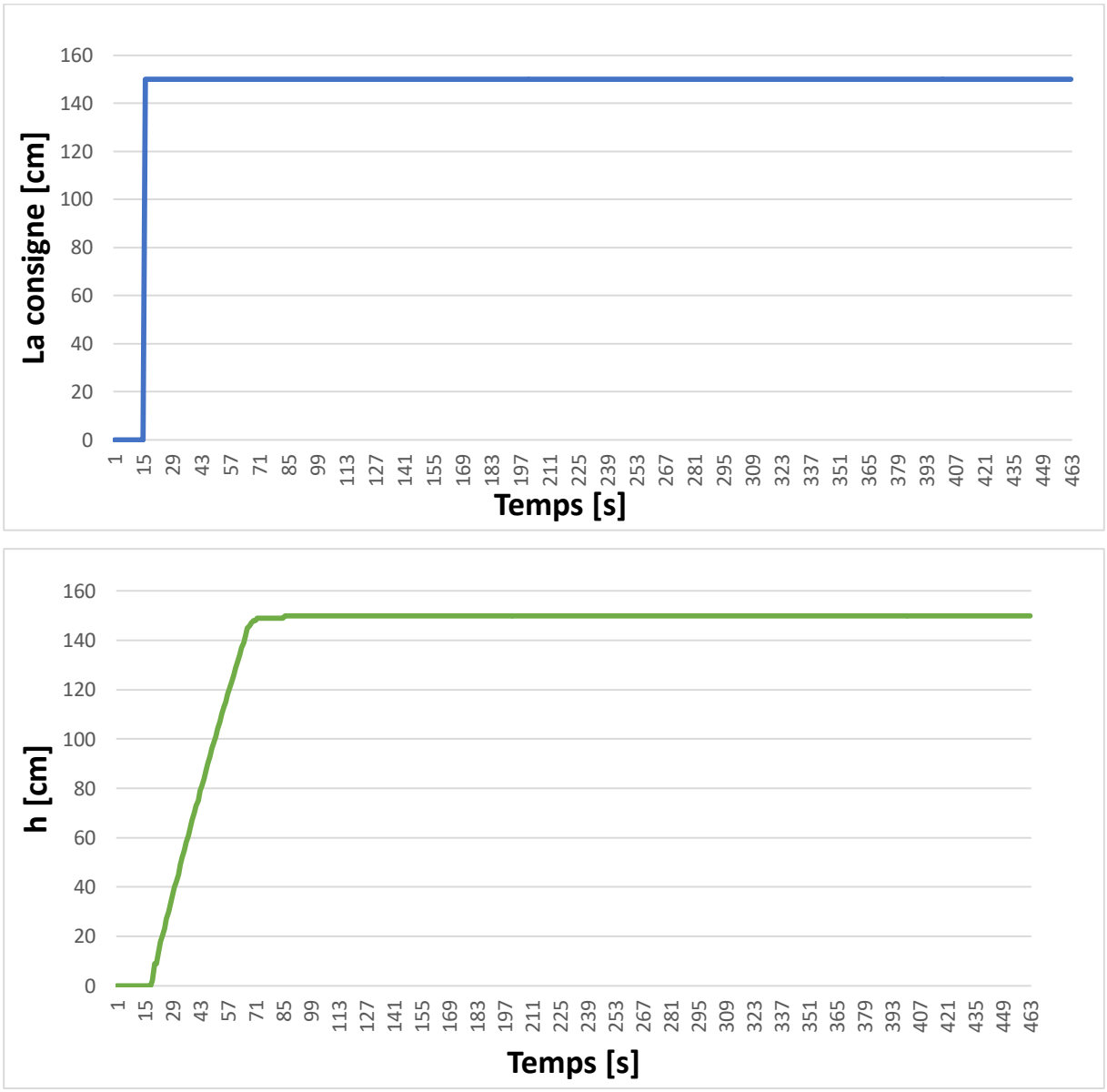


Figure 4.39 : Résultats de simulation avec SCL

4.5.4 Comparaison

Les résultats de simulation du programme obtenu sous STEP 7 avec PLCSIM sont identiques lors l'utilisation de bloc PID FB41 de STEP 7 (avec Ladder) ou bien le bloc que nous avons programmé (avec SCL).

D'après ces résultats on remarque que le système est stable et la sortie de ce dernier converge vers la consigne désirée.

Donc l'objectif de régulation de niveau de ce système est atteindre.

4.6 Conclusion

Dans ce chapitre, nous avons présenté le système à réservoir du logiciel Factory I/O, les parties constituant le programme Step 7 utilisé pour la régulation du niveau de ce système à réservoir, et la partie IHM utilisée pour visualiser et contrôler ce système. A la fin nous avons présenté des simulations de ce système dans le logiciel WinCC flexible.

Conclusion Générale

Conclusion générale

Dans ce mémoire, nous nous sommes intéressés à la régulation PID et la supervision par API du niveau d'un réservoir de stockage cylindrique simulé dans le logiciel Factory I/O. Notre travail consiste à élaborer un programme de régulation PID pour les automates S7-300 de siemens en utilisant le logiciel Step 7 et de créer une interface IHM pour la supervision du processus sous WinCC flexible.

Nous avons présenté en premier lieu les diverses formes des réservoirs utilisés dans le secteur industriels pour le stockage des liquides. En second lieu, nous avons passé en revue les différentes formes des régulateurs PID avec leurs méthodes d'ajustement des paramètres. Par la suite, nous avons présentés de façon succincte les logiciels utilisés pour la programmation et la supervision des API siemens, en l'occurrence, le logiciel Step 7 et le logiciel WinCC flexible 2008. A la fin, nous avons présenté le programme de régulation PID réalisé sous Step 7 avec le langage Ladder et le langage SCL (texte structuré) avec un système de supervision sous WinCC flexible, permettant un suivi de l'évolution du processus en temps réel. Les résultats de simulation obtenus valident bien notre programme ainsi que l'interface de supervision proposée.

Concernant les perspectives, nous proposons l'implémentation de la régulation floue en utilisant un automate programmable industriel.

Bibliographie

[1] K. Kukla, **Optimal Control of Industrial Storage Tanks**, Projet de fin d'étude, Slovak University of Technology In Bratislava, Faculty of Chemical and Food Technology, 2018.

[2] N. Ghers, **Approche fiabiliste dans la vérification du dimensionnement de réservoir de stockage,** Projet de fin d'étude, Université Badji Mokhtar Annaba, 2015/2016.

[3] S. Sekhsoukh, K. Oukili, E**tude d'une boucle de régulation de niveau implémentation du régulateur et réglage du procédé**, Projet de fin d'étude, école supérieur de technologie, 2010/2011.

[4] A. Y. Kadri, **Régulation automatique**, Support cours, Université de Ouargla, 2013/2014.

[5] S. Labiod, **PID cours synthèse correcteur classique**, Support cours, Université de Jijel.

[6] K. Amoura, **Méthode basée sur le dépassement de la réponse indicielle pour le réglage des contrôleurs PID**, Projet de fin d'étude, Université Mouloud Mammeri De Tizi-Ouzou, 2016 /2017.

[7] H. Garnier, **Synthèse d'un correcteur PID numérique par transposition du PID analogique**, Support cours, Université de Lorraine,

[8] E. Ulamine, A. Hammouali, C**onception de régulation de niveau avec un automate programmable**, Mémoire de fin d'étude, 2018/2019.

[9] B. K. Kangni, I**ntroduction des automates programmables industriels sur les locomotives diesels électriques a l'O.T.P : incidences économiques et technique**, Projet de fin d'étude, Université cheikh Anta Diop, 1992.

[10] L. Bergougnoux, **API automate programmable industriel,** Poly Tech Marseille, Support cours, 2004/2005.

[11] L. Andjouh, R. Touati, **Automatisation et supervision de la fosse de relevage de la raffinerie d'huile au niveau du complexe agroalimentaire Cevital**, Projet de fin d'étude, Université de Bejaia, 2012/2013.

[12] A. Tensaout, T. Youcef Khoudja, C**onception d'une régulation de niveau avec un automate programmable**, Projet de fin d'étude, Université de Bejaia, 2014/2015.

[13] SIMATIC, **Automate programmable S7-1200**, Manuel, 06 /2015.

[14] SIMATIC, **S7-1500, ET 200MP système d'automatisation**, Manuel ,11 /2019.

[15] SIMATIC, P**rogrammer avec STEP-7**, Manuel ,05/2010.

[16] SIMATIC, **Logiciel de base pour S7-300/400 Régulation PID**, Manuel.

[17] SIMATIC, **WinCC flexible, Brochure**, Manuel, 03/2010.

[18] SIMATIC, **WinCC flexible 2008 Compact/Standard/Advanced**, Manuel, 07 /2008.

[19] SIMATIC, **Outils d'ingénierie S7-PLCSIM V5.4**, Manuel, 07/2011.