Technicien Spécialisé

Génie électrique

Électromécanique des Systèmes Automatisés

Office de la Formation Professionnelle et de la Promotion du Travail

Manuel de cours

Module 23

Réseaux et systèmes de supervision et de contrôle commande

Edition 2021

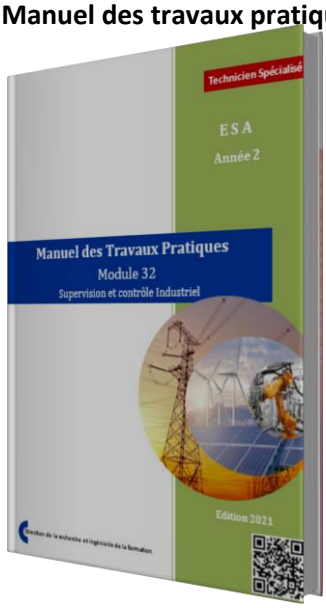
Direction de la Recherche et Ingénierie de la Formation

Avant-propos

Les manuels de cours, de travaux pratiques et le guide e-learning sont téléchargeables à partir de la plateforme e-learning moyennant le scanne des codes QR suivants :

Manuel de cours

Manuel des travaux pratiques



Guide e-learning

SOMMAIRE

COMPETENCES-CIBLES ET OBJECTIFS OPERATIONNELS

Module 23 : Réseaux et systèmes de supervision et de contrôle commande

Code : GEESA-23 **Durée : 60 heures**

ENONCE DE LA COMPETENCE

Utiliser et dépanner les réseaux et systèmes et de supervision et de contrôle commande

CONTEXTE DE REALISATION

- Individuellement
- À partir de :
  - Directives ;
  - Manuels et Fiches techniques ;
  - Documents techniques
- À l'aide de :
  - Logiciel de supervision
  - Architecture réseau
  - Notice d'utilisation des systèmes de supervision et contrôle commande
  - Station de travail
  - Câbles réseaux
  - Outils de diagnostic réseaux

CRITÈRES GÉNÉRAUX DE PERFORMANCE

- Pertinence de la terminologie utilisée.
- Respect des consignes de sécurité
- Conformité au cahier des charges
- Maintenance correcte des réseaux et application de supervision
- Fonctionnement correcte des réseaux et application de supervision
- Amélioration notable de fonctionnement des réseaux et application de supervision

ÉLEMENTS DE LA COMPETENCE	CRITÈRES PARTICULIERS DE PERFORMANCE
A. Mettre en œuvre une application de supervision	• Informations correctes sur l'application de supervision • Paramétrage correcte des interfaces de l'application de supervision • Simulation correcte de fonctionnement de l'application de supervision • Interprétation adéquate des résultats de la simulation de l'application de supervision
B. Manipuler un logiciel de supervision	• Paramétrage approprié des interfaces de l'application de supervision • Optimisation précise de l'archivage de l'application de supervision • Modifications correctes sur l'application de supervision • Amélioration notable sur l'application de supervision
C. Dépanner les réseaux terrain	• Utilisation appropriée des outils de dépannage des réseaux de terrain • Utilisation correcte des logiciels de dépannage des réseaux de terrain • Communication correcte des réseaux de terrain • Diagnostic correcte des réseaux de terrain
D. Dépanner une application de supervision	• Paramétrage correcte des applications de supervision • Utilisation correcte des outils de dépannage des applications de supervision • Utilisation correcte des logiciels de supervision • Fonctionnement correcte de l'application de supervision • Utilisation correcte des applications tierce aides au dépannage et au diagnostic des applications de supervision

Chapitre I

INTRODUCTION A LA SUPERVISION INDUSTRIELLE

1. INTRODUCTION A LA SUPERVISION INDUSTRIELLE

La **supervision industrielle**, consiste à **surveiller l'état de fonctionnement d'un procédé** pour l'amener et le maintenir à son point de fonctionnement optimal.
Née du besoin d'un **outil de visualisation** des processus industriels, dans un contexte économique de productivité et de flexibilité, la supervision a bénéficié d'une avancée technologique exceptionnelle.
À ses débuts, elle se composait d'un grand tableau mural représentant la vision des opérateurs du processus industriel. Rapidement, avec l'essor informatique, les voyants ont été remplacés par des écrans et des claviers. Le but restait le même : **contrôler et commander un processus industriel**.
La supervision industrielle est utilisée par de nombreux procédés, soit pour la surveillance d'équipements ou de locaux, on parlera alors de GTC (gestion technique centralisée), soit comme SNCC (systèmes numériques de contrôle-commande).

1.1 Définitions et fonctionnalités

Pour donner suite à l'automatisation industrielle, l'opérateur humain a été contraint de **conduire** ou de **superviser** des machines automatisées.
L'opérateur chargé de conduire une installation automatisée doit impérativement disposer en temps réel d'une visualisation de l'état et de l'évolution des paramètres du processus qui lui permettre de prendre rapidement les décisions appropriées à ses objectifs. Cette fonction d'assistance à l'opérateur humain est appelée superviseur.
Un superviseur est souvent constitué :

- D'un module d'**acquisition** et de **traitement** des signaux physiques du procédé ;
- D'un module de **commande temps réel** qui élabore les commandes en fonction des consignes, des signaux acquis et selon des modèles de commande prédéfinis ;D'un module de **contrôle** qui permet de surveiller la commande, l'évolution du procédé, de déclencher des procédures de **sécurité** (arrêts d'urgence) ou de prévenir l'opérateur d'une situation anormale ;
- D'un module de **visualisation-stockage**, qui permet d'obtenir et de mettre à la disposition des opérateurs des éléments d'évaluation du procédé par ses valeurs instantanées et historiques.

1.2 Fonctions de la supervision

En situation normale, le système de supervision présente, sur les **synoptiques**, une ou plusieurs vues de synthèse sur le système industriel, et une ou plusieurs vues spécialisées sur la phase de l'activité principale en cours et sur les éléments du système concerné.

Vue synoptique : permet à l'opérateur d'interagir avec le processus et de visualiser le comportement normal.

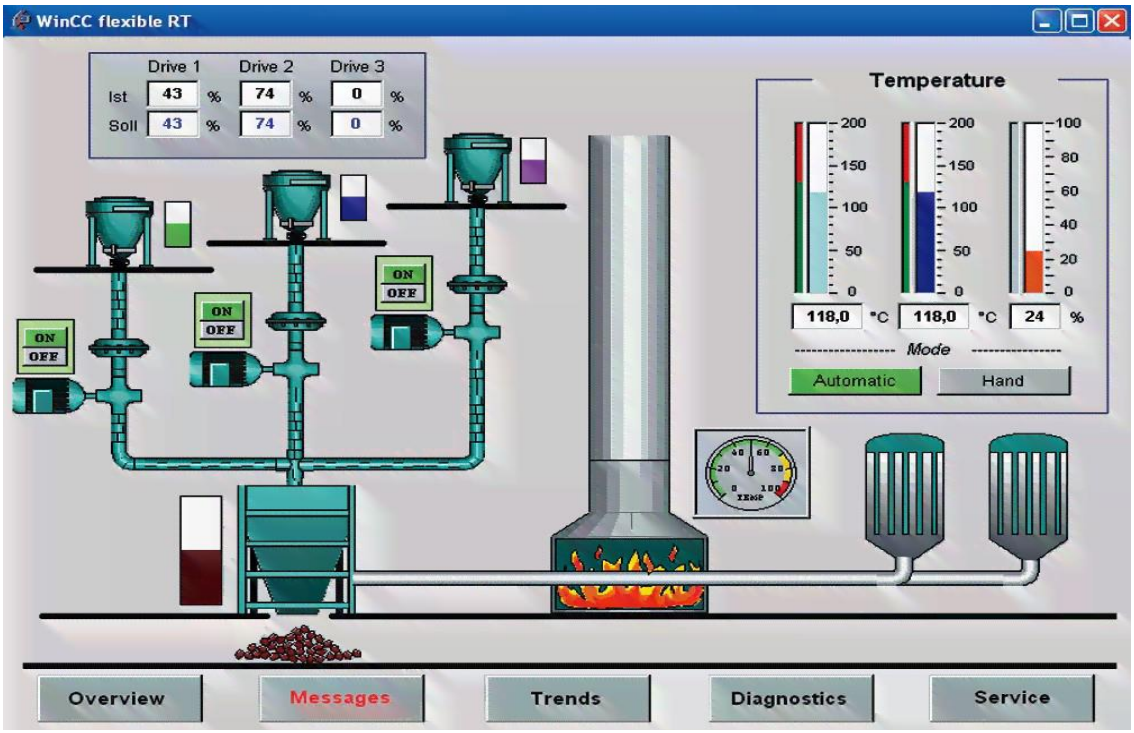


Figure 1.1 : Vue synoptique d'une application de supervision

Les modules de contrôle du système automatisé génèrent **des alarmes** selon une hiérarchie propre à chaque système. Un journal enregistre tous les événements significatifs survenus sur le système pendant que les écrans de contrôle de l'opérateur retransmettent les alarmes. Le degré d'élaboration de ces alarmes dépend beaucoup des systèmes et de l'effort de modélisation préalable à l'automatisation du système.

Le module de gestion des alarmes calcule en temps réel les conditions de déclenchement des alarmes, affiche l'ensemble des alarmes selon des règles de priorité, donne les outils de gestion depuis la prise en compte jusqu'à la résolution complète, assure l'enregistrement de toutes les étapes de traitement de l'alarme.

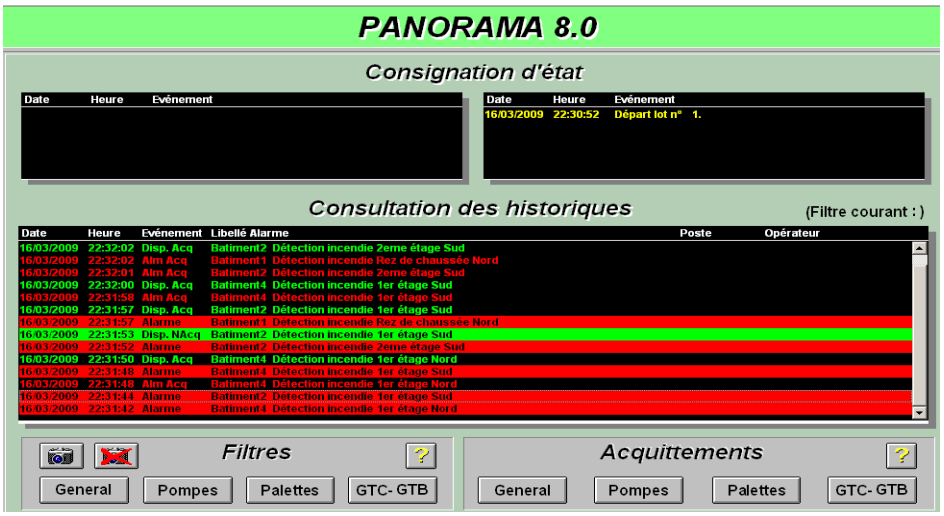


Figure 1.2 : Le module de gestion des alarmes

La communication homme-machine doit être particulièrement étudiée pour rendre efficace l'interaction entre le système d'aide à la supervision et l'opérateur. L'interface de supervision permet de suivre sur un même écran l'évolution de plusieurs variables analogiques. Ces vues sont très utiles à l'opérateur de conduites car elles lui permettent **d'anticiper sur l'évolution du procédé**. Ces courbes donnent une représentation graphique de différentes données du processus et donnent aussi les outils d'analyse des variables historisées.

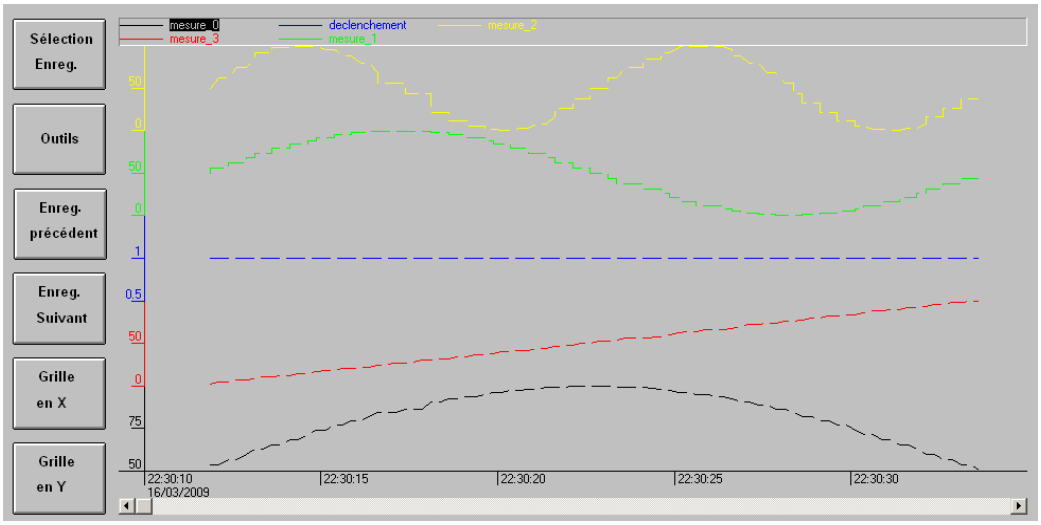


Figure 1.3 : Courbes

1.3 Architecture de la supervision

La supervision est d'un niveau supérieur et qui superpose à la boucle de commande, elle assure les conditions d'opérations pour lesquelles les algorithmes d'estimation et de commande ont été conçus. Parmi les taches principales de la supervision se trouve la surveillance, l'aide à la décision, le diagnostic et la détection.

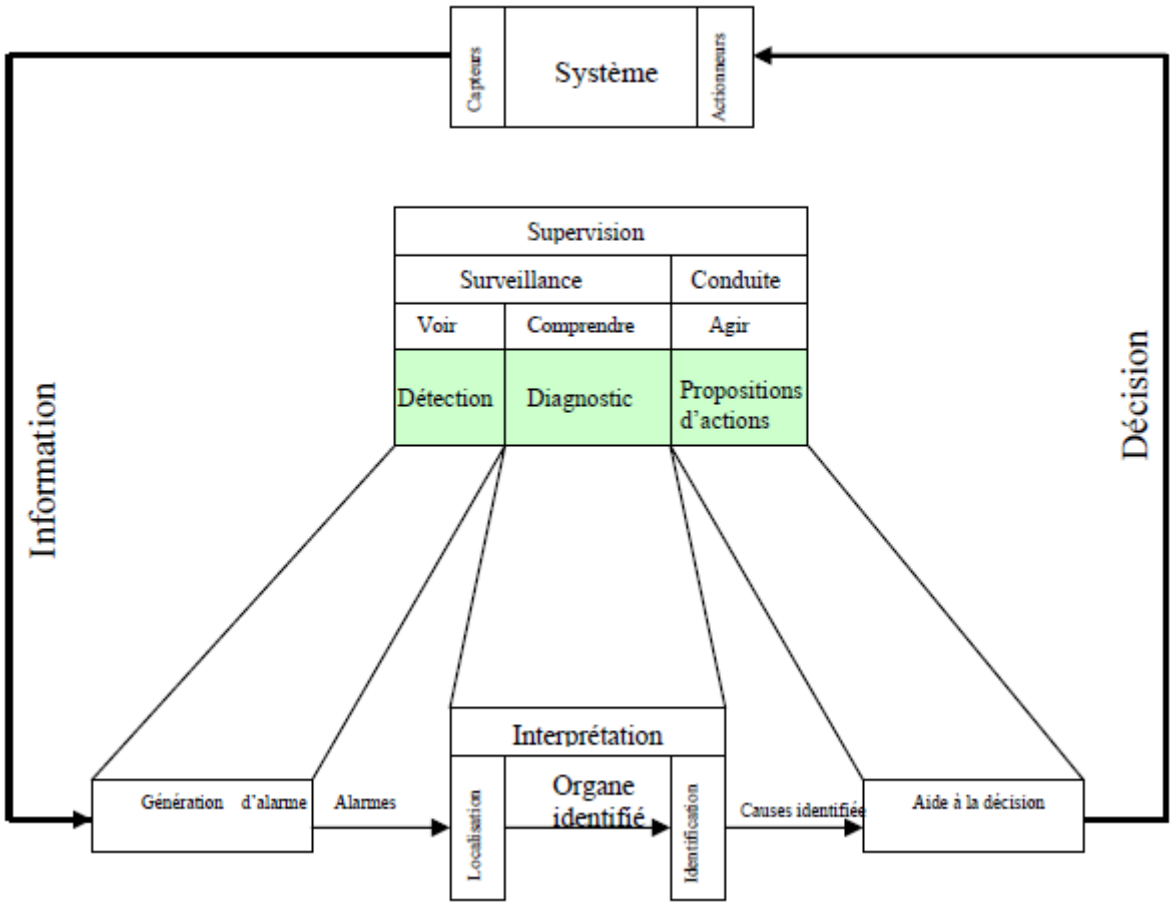


Figure 1.4 : Architecture générale d'un système de supervision

1.4 Techniques de la supervision

Pour concevoir un système de supervision on a besoin de maîtriser les techniques suivantes :

1.4.1 Acquisition des données (Data Acquisition)

L'acquisition de données est la première étape de la supervision, tel que, elle consiste à recueillir, à valider et à assurer l'acheminement des informations sur l'état du système jusqu'au poste de pilotage, cette tâche, est exécutée sans interruption et à chaque instant, ces opérations impliquent l'utilisation des capteurs permettant de mesurer les différentes variables du processus.

1.4.2 Surveillance

La surveillance utilise les données provenant du système pour représenter l'état de fonctionnement puis en détecter les évolutions. La surveillance intervient en phase d'exploitation bien qu'elle soit prise en compte dès la phase de conception. Elle sert à filtrer les signaux et les événements issus du procédé et de la commande afin d'établir l'état du système. En fonctionnement normal, elle communique des comptes rendus filtrés à la commande. Elle permet également de détecter et diagnostiquer les fautes et les erreurs dans le système. En cas de dysfonctionnement, elle en informe le module de maintenance et le module de supervision.

1.4.3 Diagnostic

Cette étape consiste à partir des défauts détectés, de localiser l'élément défaillant et d'identifier la cause qui a provoqué ce défaut.

- *Localisation* : Cette étape s'exécute juste après qu'il ait une détection d'un défaut, elle consiste à repérer et à isoler l'élément défaillant et préparer ainsi la tâche à la prochaine étape qui est l'identification.
- *Identification* : Ici, on cherche à identifier les causes précises de cette anomalie et à réparer le dysfonctionnement. Les informations ainsi obtenues sont fournies au service de maintenance.

1.4.4 Aide à la décision

L'aide à la décision consiste à aider l'opérateur à prendre la bonne décision devant toute situation, et cela en proposant une liste d'actions qui pourraient restaurer les grandeurs optimales du système.
Dans un système d'aide à la décision, l'opérateur est toujours maître de la situation, car le système d'aide à la décision n'agit jamais, il informe et conseille seulement, de ce fait le système ne calcule pas une valeur précise, mais propose plutôt quel moyen d'action doit être exécuté, par exemple, il pourrait conseiller de changer un point de référence d'une boucle de commande parce qu'un capteur a une dérive ou de changer la commande manuelle parce que le régulateur travaille hors de son domaine de stabilité.

1.4.5 La maintenance

La maintenance est l'étape qui intervient généralement après l'étape de prise de décision elle consiste à maintenir ou à restaurer les performances des composants ou du système d'une façon globale, pour l'accomplissement de sa tache requise, ces activités sont une combinaison d'activités techniques, administratives et de gestion.
Lorsqu'au cours d'une tâche préventive un composant interne du matériel est trouvé ou jugé défaillant, sa réparation ou son remplacement doit être considéré comme de la maintenance corrective. S'il est trouvé non défaillant mais dégradé, même au-delà de la valeur de défaillance potentielle, sa réparation ou son remplacement est de nature préventive.

- **Maintenance préventive** : L'objectif de la maintenance préventive est de diminuer la probabilité de défaillance ou la dégradation d'un composant qui pourrait nuire à sa fonction requise, ce type de maintenance intervient si une durée de vie d'un composant est expirée (maintenance systématique) ou si ce composant se trouve dans un état de dégradation significatif et qui pourrait provoquer une défaillance sur le système qui le rendra incapable d'accomplir sa fonction requise (maintenance conditionnelle).
- **Maintenance corrective** : La maintenance corrective consiste en toute tache réalisée pour rétablir le bon fonctionnement du système, que ce soit d'une façon définitive ou d'une façon temporaire, et cela après qu'un défaut ait lieu.

1.5 Logiciels de supervision

Les logiciels de supervision sont une classe de programmes applicatifs dédiés à la production dont les buts sont :

- La collecte d'informations en temps réel sur des processus depuis des sites distants (machines, ateliers, usines...) et leur archivage

- la visualisation de l'état et de l'évolution d'une installation automatisée de contrôle de processus, avec une mise en évidence des anomalies (alarmes)
- l'assistance de l'opérateur dans ses actions de commande du processus de production (interface IHM dynamique...)
- l'aide à l'opérateur dans son travail et dans ses décisions (propositions de paramètres, signalisation de valeurs en défaut, aide à la résolution d'un problème ...)
- fournir des données pour l'atteinte d'objectifs de production (quantité, qualité, traçabilité, sécurité...)

⇨ Quelques superviseurs commerciaux:

Schneider Electric	Monitor Pro	Codra	Panorama P2, Panorama E2
AREAL	TopKapi	ARC Informatique	PcVue, PlantVue
ELUTIONS	ControlMaestro, Wizcon	SIEMENS	SIMATIC WinCC Version 7
ICONICS	Genesis 32	Wonderware	InTouch, InControl

1.6 Supervision dans un environnement SCADA

1.6.1 Définition du SCADA

SCADA est un acronyme qui signifie le contrôle et la supervision par acquisition de données (en anglais: Supevisory Control And Data Acquisition). Le système SCADA collecte des données de diverses appareils d'une quelconque installation, puis transmit ces données à un ordinateur central, que ce soit proche ou éloigné, qui alors contrôle et supervise l'installation.

Le système SCADA fonctionne par l'acquisition de données provenant de l'installation, ces dernières sont affichées sur une interface graphique sous un langage très proche de langage humain, ces opérations sont exécutées en temps réel, ainsi les systèmes SCADA donnent aux opérateurs le maximum d'information pour une meilleure décision, ils permettent un très haut niveau de sécurité, pour le personnels et pour l'installation et permettent aussi la réduction des coûts des opérations, les avantages qu'offre le SCADA sont obtenus avec la combinaison des outils softs et hard.

1.6.2 Architecture du SCADA

SCADA entoure un transfert de données entre le Serveur (MTU, master terminal units) et une ou plusieurs unités terminales distantes (Remote Terminal Units RTUs), et entre le Serveur et les terminaux des opérateurs, la Figure 5 représente un schéma sur l'architecture d'un réseau SCADA qui utilise des routeurs pour joindre le poste de pilotage par le billet de l'Internet. Les logiciels de supervision sont une classe de programmes applicatifs dédiés au contrôle de processus et à la collecte d'informations en temps réel depuis des sites distants (ateliers, usines), en vue de maîtriser un équipement (machine, partie opérative).

Les éléments hardware assurent la collecte des informations qui sont à disposition du calculateur sur lequel est implanté le logiciel de supervision, le calculateur traite ces données et en donne une représentation graphique réactualisée périodiquement, le système SCADA enregistre les événements dans des fichiers ou les envoie sur une imprimante, par mail..., ainsi le système surveille les conditions de fonctionnement anormal et génère des alarmes.

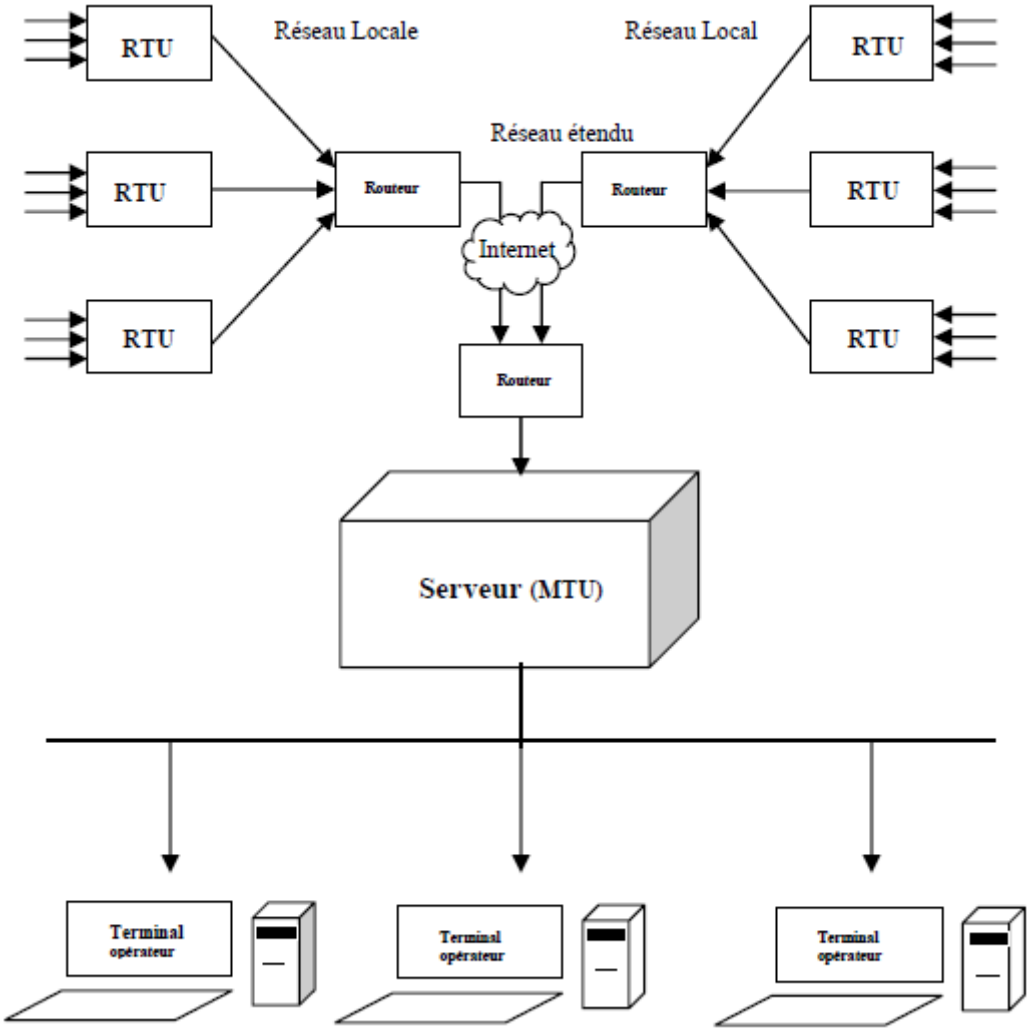


Figure 1.5 : Architecture de la supervision dans un environnement SCADA

1.6.3 Avantages du SCADA

Parmi les avantages du SCADA on retrouve :

- Le suivi de près du système ; voire l'état du fonctionnement de procédé dans des écrans même s'il se situe dans une zone lointaine.
- Le contrôle et l'assurance que toutes les performances désirées sont atteintes ; de visualiser les performances désirées du système à chaque instant, et s'il y aurait une perte de performance, une alarme se déclenchera d'une manière automatique pour prévenir l'opérateur.
- Produire une alarme lorsqu'une faute se produit et visualise même la position où se situe la faute et l'élément défectueux, ce qui facilite la tâche du diagnostic et de l'intervention de l'opérateur.
- Donne plusieurs informations sur le système ainsi aide l'opérateur à prendre la bonne décision, et ne pas se tromper dans son intervention.
- Diminue les taches du personnel en les regroupant dans une salle de commande.
- Elimination ou réduction du nombre de visite aux sites éloignés ; avec une interface graphique, on peut suivre l'état de l'installation à chaque instant, ainsi on n'aura pas besoin de faire des visites de contrôle.

1.6.4 Interfaces graphiques du SCADA

Les interfaces graphiques sont un outil très important pour le bon déroulement de la procédure d'aide à la décision, elles sont le seul point d'interaction entre l'opérateur et les algorithmes d'aide à la décision, ainsi, elles aident l'opérateur dans sa tâche d'interprétation et de prise de décision, en lui offrant une très bonne visibilité sur l'état et l'évolution de l'installation, avec l'affichage en différentes couleurs des résidus, des alarmes et des propositions sur l'action à entreprendre, La **Figure 1.6** représente une interface graphique de la supervision d'un générateur de vapeur :

1.6.5 Fonctionnalités Temps Réel

La notion temps réel est devenue très importante et indispensable dans la procédure de surveillance et de supervision en générale, elle permet de faire le rafraichissement des signaux à chaque instant, ce qui permet de suivre l'évolution de l'état du système d'une façon continue.

1.7 Conclusion sur le rôle de la supervision

La supervision joue un très grand rôle dans la sécurité du personnel ainsi que sur l'environnement, et ce, en détectant la moindre dégradation qui pourrait affecter le bon fonctionnement du système, voir même des explosions et des dommages matériels et humains.
L'arrêt d'un système industriel provoque des dommages énormes sur l'économie de l'entreprise, alors on peut conclure que le rôle de la supervision ne se limite pas à la sécurité mais aussi à la continuité de la rentabilité de l'entreprise, autrement dit à la survie de l'entreprise.

Le système SCADA est un outil qui permit de réaliser une supervision à distance, c'est-à-dire, que l'installation à superviser pourrait se trouver à des milliers de kilomètres du poste de pilotage, ce type

de supervision est très utile pour les industries à hauts risques, telles que les industries chimiques et nucléaires car il évite des pertes humaines si jamais un accident survient et aussi il réduit énormément le nombre de visite au site.

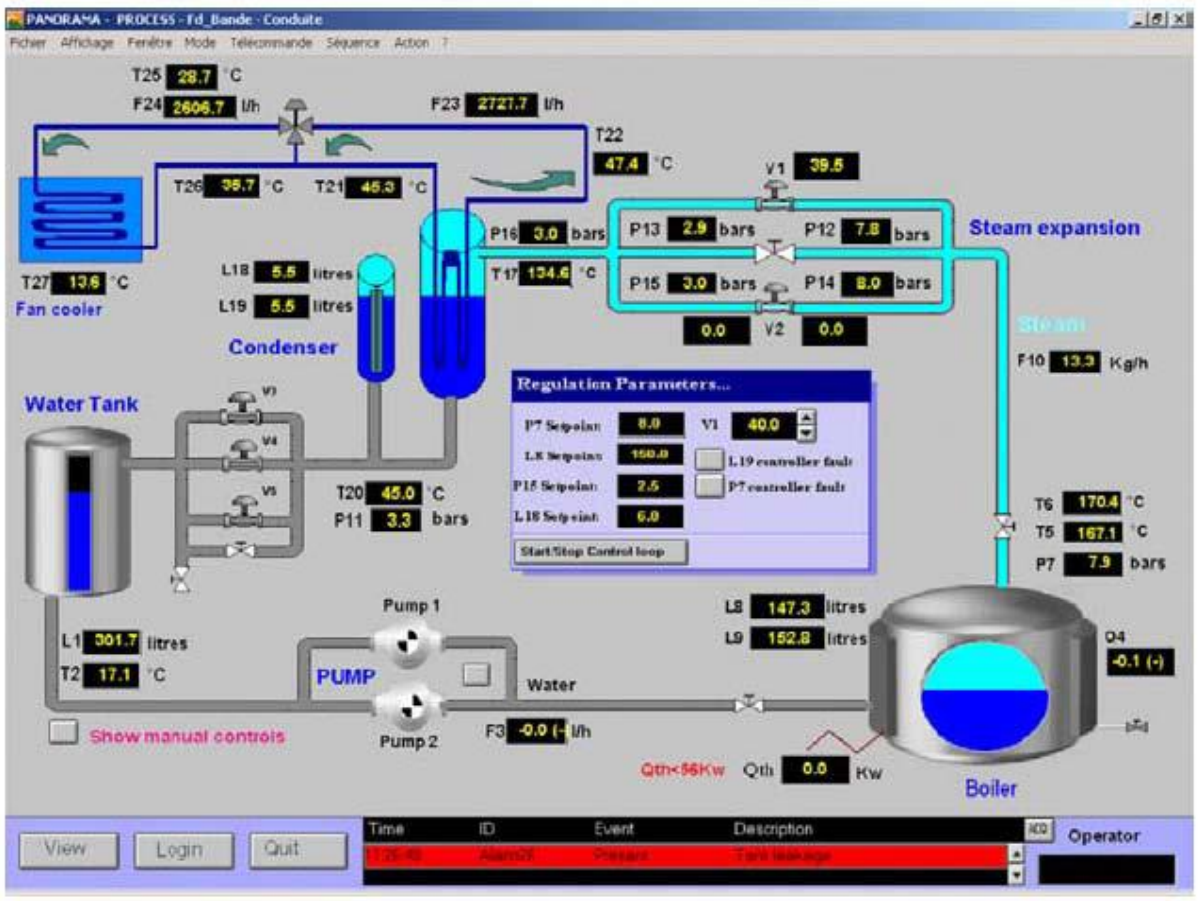


Figure 1.6 : Interface graphique de supervision d'un générateur de vapeur

Chapitre II

INTERFACE HOMME/MACHINE (IHM)

APPLICATION AUX SYSTEMES INDUSTRIELS

2. INTERFACES HOMME/MACHINE(IHM) : APPLICATIONS AUX SYSTEMES INDUSTRIELS

Chaque processus industriel comporte une interface de communication homme-machine permettant de piloter les modes de marche (marche, arrêt, auto, manuel, pas à pas...) et d'assurer la surveillance de l'état du processus.
Selon la nature du processus (continu, discret) et sa complexité, l'interface est plus ou moins sophistiquée.
L'élément de base de l'interface de communication homme machine est une console opérateur comportant un écran graphique couleur, un clavier et une imprimante.

2.1 Structure fonctionnelle

L'interface de communication homme-machine d'un système de supervision comporte généralement trois grandes fonctions :

- Surveillance et conduite du processus ;
- Archivage de données ;
- Gestion de données.

2.1.1 Surveillance et conduite du processus

Cette fonction est assurée à l'aide de vues de groupe, de réglage, de vues synoptiques, de tendance et d'alarmes.
La plupart des SNCC offrent la possibilité de décrire des liens entre les différents types de vues, ce qui permet une navigation entre ces vues avec une seule opération de sélection à l'aide du clavier ou de la souris.

2.1.1.1 Vue de groupe

Ces vues (figure 2.1) représentent les différentes variables concernant chaque boucle d'un groupe sous forme de barreaux et de valeurs numériques. Chaque boucle occupe une case contenant les informations suivantes :

- la valeur de la mesure, visualisée sous forme de barreau et sous forme numérique en unités physiques ;
- la valeur de la consigne, sous forme de barreau et sous forme numérique ;
- la valeur du signal de commande, sous forme de barreau et sous forme numérique dans une échelle 0-100 % ;
- l'état de la consigne (interne ou externe) ;
- l'état de la commande (manuelle/automatique).

Des renseignements complémentaires comme la valeur des seuils d'alarme ou les limites du signal de commande sont, soit présentés en permanence, soit appelés par des touches spécifiques du clavier.

L'opérateur peut, après sélection d'une des boucles à l'aide d'une touche affectée à la zone d'écran correspondante, effectuer des commandes sur cette boucle par l'intermédiaire de touches spécifiques d'incrémentation-décrémentation ou d'un clavier décimal.
Un effet de loupe (zoom) sur la boucle sélectionnée peut également être obtenu par l'intermédiaire du clavier, c'est la **vue de réglage.**

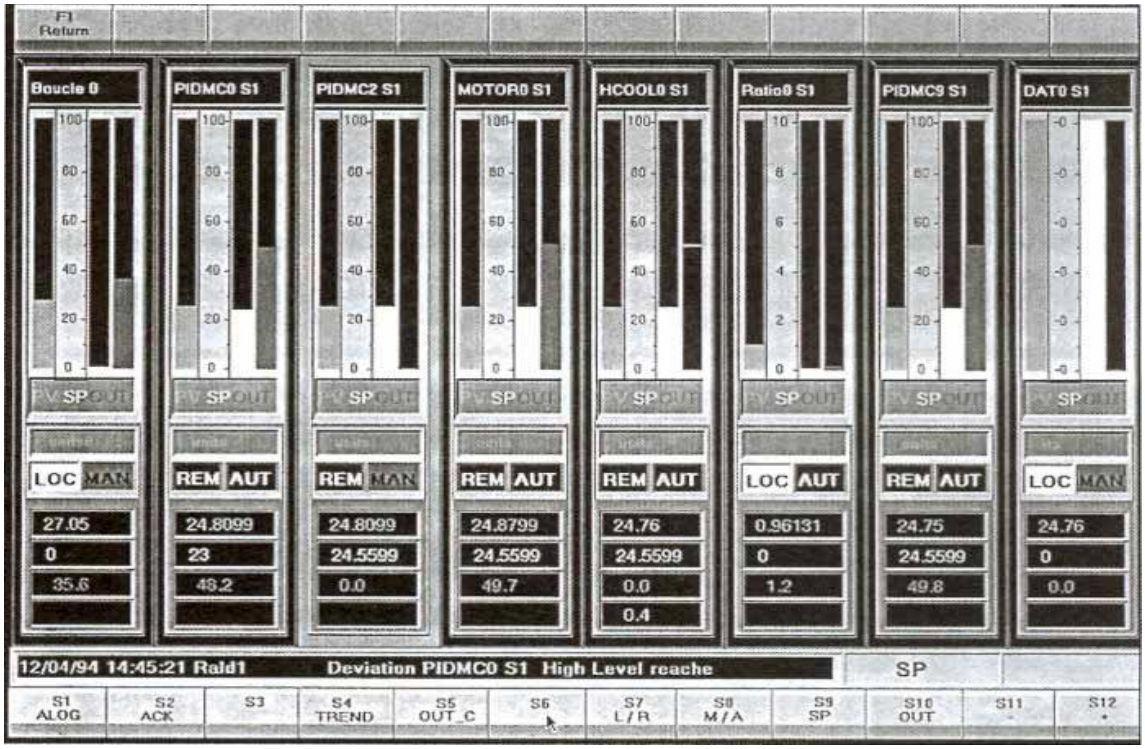


Figure 2.1 La vue de groupe

La figure précédente montre la vue de groupe qui permet de surveiller un ensemble de régulateurs et, après sélection d'une des boucles au clavier, de modifier son mode de marche (manuel/automatique), sa consigne ou sa sortie

2.1.1.2 Vue de réglage

Ces vues (figure 2.2) reprennent les informations d'une zone de la vue de groupe avec des renseignements complémentaires, essentiellement les coefficients de réglage du régulateur, qui sont alors modifiables à l'aide du clavier décimal.

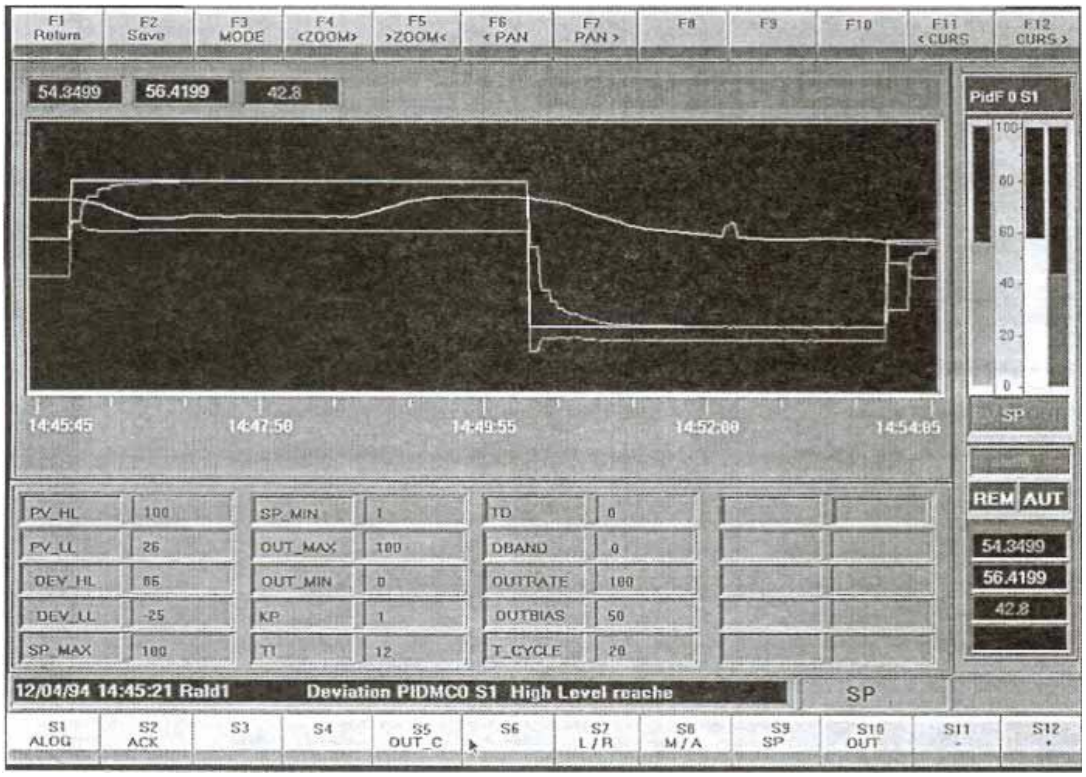


Figure 2.2 – La vue de réglage

2.1.1.3 Vue synoptique

Ces vues (figure 2.3) sont organisées sous la forme d'une **arborescence** qui présente le processus à différents niveaux de détails.
Chaque synoptique comporte un certain nombre de variables (états, mesures) animées sous forme d'icônes choisies dans une bibliothèque. La sélection d'une icône (à l'aide de la souris, boule roulante...) appelle une fenêtre qui fournit des détails complémentaires sur la variable associée et permet sa commande si c'est une sortie.

La vue de réglage précédente donne accès aux coefficients de réglage d'un régulateur (ex. : Kp , Ki , Kd , etc.) et à une vue de tendance de ses variables principales (mesure, consigne, sortie)

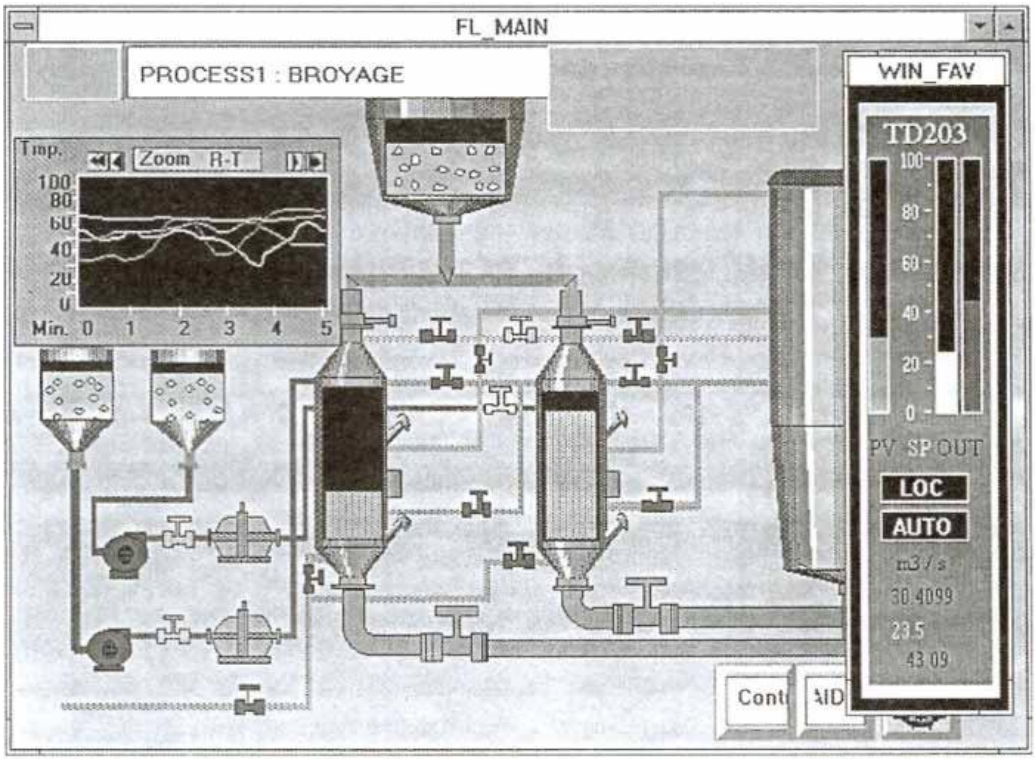


Figure 2.3 Vue synoptique

2.1.1.4 Vues de tendance et historiques

Les vues de tendance (figure 2.4) assurent la fonction remplie antérieurement par les enregistreurs papiers multipistes. Elles permettent de suivre sur un même écran l'évolution de plusieurs variables analogiques. Ces vues sont très utiles à l'opérateur de conduites car elles lui permettent d'**anticiper sur l'évolution du procédé.** Certains systèmes offrent la possibilité d'intégrer des « minivues de tendance » dans les synoptiques, ce qui permet d'éviter des commutations d'images, très néfastes en période d'incident.

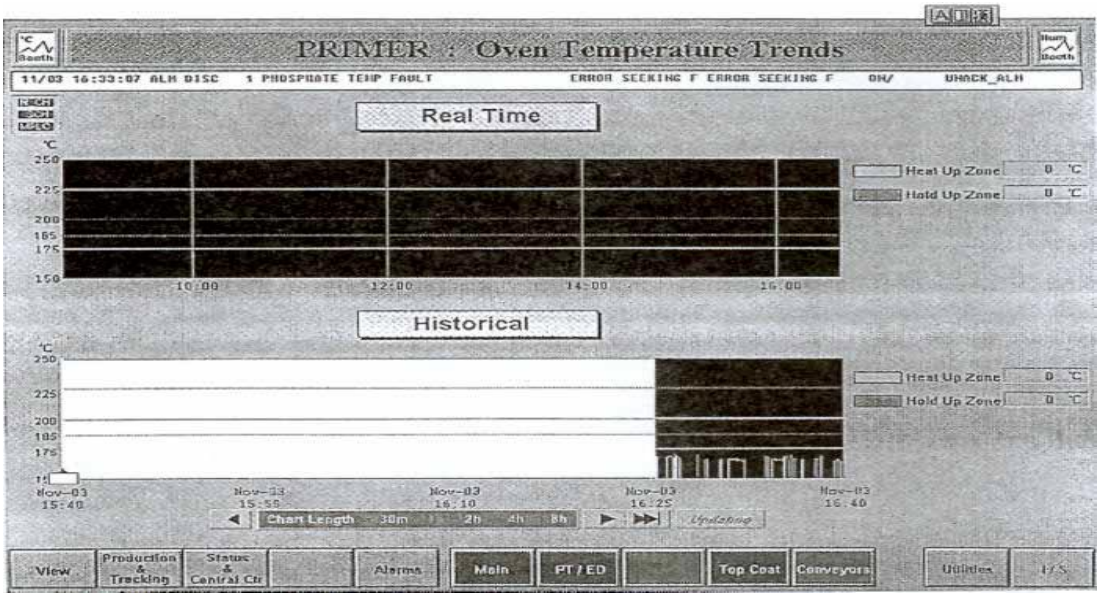


Figure 2.4 – Vue de tendance

2.1.1.5 Vues d'alarmes

Ces vues (figure 2.5) présentent les alarmes apparues sur le processus classées de différentes manières :

- par ordre chronologique ;
- par niveau de priorité ;
- par domaine (sous processus) ;
- par nature (thermique, mécanique...).

L'édition de ces alarmes s'effectue suivant un code de couleurs, par exemple :

Alarme présente non acquittée →rouge clignotant

Alarme présente acquittée → rouge fixe

Alarme disparue → blanc

Une édition *au fil de l'eau* des alarmes s'effectue en parallèle sur l'imprimante associée au poste opérateur. Les vues d'alarmes ont très peu évolué depuis les premières générations de SNCC. La détection des alarmes reste rustique (comparaison d'une mesure à un seuil). Le niveau de priorité associé est généralement déclaré de manière statique et peu de systèmes offrent la possibilité de le faire évoluer par programme.

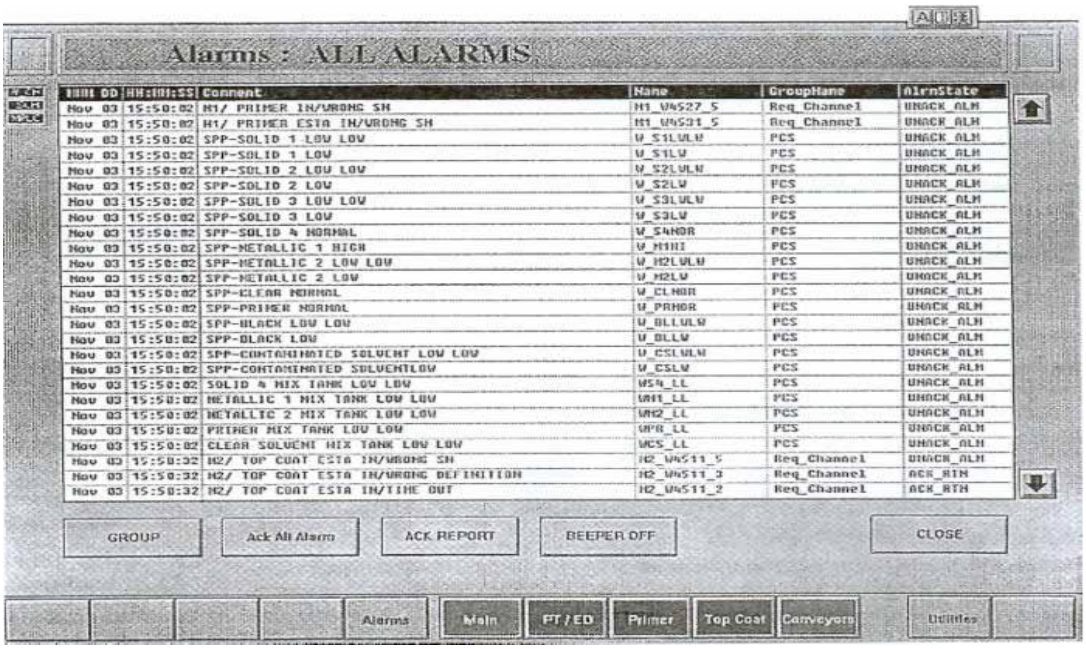


Figure 2.5 – Vue d'alarmes

2.1.2 Archivage des données

Cette fonction a pour but de mémoriser des informations sur le fonctionnement du procédé qui seront mises à disposition d'une fonction gestion de données locale à l'interface homme-machine ou d'un système informatique de niveau supérieur.

Les informations archivées sont de type TOR ou analogique.

Elles sont stockées dans la base de données de la station d'interface homme-machine. Elles sont utilisées en interne pour l'affichage de courbes et l'édition de journaux, et elles sont accessibles à un système tiers à l'aide de requêtes SQL.
La fonction archivage est très importante pour l'analyse de défauts, *a posteriori*, et elle est obligatoire dans certains domaines comme l'agroalimentaire et la pharmacie où la traçabilité des produits est exigée (rappelons que la traçabilité est définie, selon la norme ISO 8402, comme l'aptitude à retrouver l'historique, l'utilisation ou la localisation d'une entité au moyen d'identifications enregistrées).

- **Informations TOR**

Ces informations, appelées en général **« événements »**, concernent :

- soit l'évolution du procédé : c'est le cas des alarmes, des changements d'état de capteurs ou d'actionneurs TOR ;
- soit les actions de l'opérateur : démarrage/arrêt d'équipement, changement de mode de marche, modification de consigne, acquittement d'alarme...

- **Informations analogiques**

Ce sont principalement des mesures qui sont soit échantillonnées en permanence à une période configurable en fonction de la dynamique du procédé, soit stockées sur événement, c'est le cas en particulier dans le domaine du « batch ».

2.1.3 Gestion des données

La fonction gestion de données réalise des traitements sur les données archivées et donne accès aux services suivants.

- **Traitements sur les événements**

Cette fonction permet, par des tris multicritères, de solutionner des événements dans la base de données d'archives et d'effectuer des calculs (nombre de manœuvres d'une vanne, temps passé dans un état donné, temps moyen d'intervention sur défaut...) et de les présenter sous forme de courbes, histogrammes...

- **Édition de journaux/bilans**

Cette fonction permet d'éditer, sur demande ou à période fixe, des journaux de bord ou bilans d'exploitation.

2.2 Structure matérielle

La structure matérielle est présentée en figure 2.6.
Les constructeurs de systèmes de supervision, qui ont tous commencé par développer leur propre base matérielle pour les stations d'interface homme-machine, utilisent maintenant les bases matérielles standards des grands constructeurs informatiques, ce qui leur permet de bénéficier de l'amélioration constante des coûts et des performances indirectes par la compétition entre les acteurs du marché.

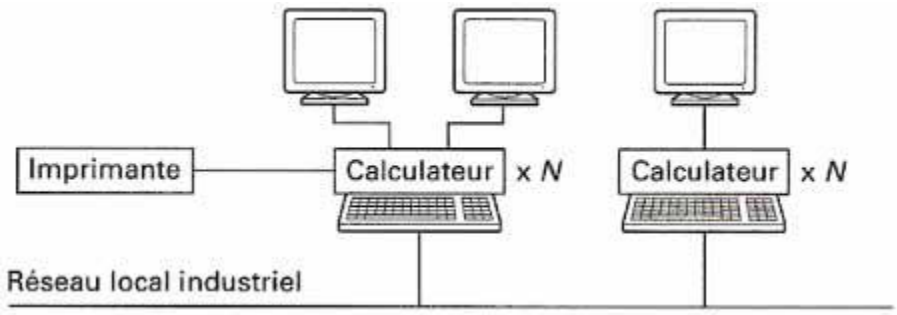


Figure 2.6 – Communication homme-machine : architecture matérielle

La base matérielle utilisée est généralement du type station de travail sous UNIX, utilisant des logiciels standards (TPC/IP, X WINDOWS...), avec accès aux services associés, architecture client/serveur, SGDB distribué ; cependant l'amélioration permanente des performances des PC (Pentium) et l'évolution de leurs systèmes d'exploitation (WINDOWS NT) vers le temps réel leur permet maintenant d'accéder aux fonctions d'interface homme-machine à des prix très compétitifs.
La plupart des fabricants de SNCC proposent maintenant une version mini ou micro de leur système de base réalisé à partir de PC pour l'interface homme-machine.
Un SNCC peut être monoposte ou multiposte, chaque poste peut être mono-écran ou multi-écrans. Les différents postes peuvent être banalisés ou affectés chacun à la conduite d'une partie de l'installation.
Les différents écrans d'un poste peuvent être dédiés à un type de vue particulier (ex : écran d'alarmes).

2.3 Structure logicielle

La structure logicielle d'une station d'interface homme-machine reste relativement classique. Elle s'appuie généralement sur les modules suivants.

- **Base de données temps réel**

Ce module contient une image des variables du procédé rafraîchie en temps réel. Il alimente les différentes tâches qui viennent y puiser les informations nécessaires pour l'édition et le traitement des données. Ce module contient des données provenant des contrôleurs de processus et également des données internes à la station, créées et utilisées par les différentes tâches.

- **Serveur de communication**

Le serveur de communication gère les échanges de données entre la station d'interface homme-machine et les contrôleurs de processus.
Ces échanges peuvent être réalisés de différentes manières :

- scrutation périodique à l'initiative de la station d'interface homme-machine ; c'est en général le cas pour toutes les variables déclarées en historique et les variables correspondant à la vue courante sur l'écran ;
- émission sur événements à l'initiative des contrôleurs de processus ; c'est le cas des alarmes et changements d'état TOR ;
- émission sur événements à l'initiative de la station d'interface homme-machine ; c'est le cas des commandes opérateur.

- **Tâche graphique**

Cette tâche assure l'affichage des vues sur l'écran et la gestion des outils de saisie (clavier, boule, souris...).
Cette tâche est très importante car c'est à travers ses caractéristiques :

- ergonomie d'utilisation ;
- temps d'affichage des images ;
- possibilités d'animation ;
- qualité du dessin ;
- nombre de couleurs ;

- **Tâche alarme**

Cette tâche prépare les données qui permettront à la tâche graphique d'afficher les différentes vues d'alarmes, et à la tâche historique d'archiver les alarmes.

- **Tâche historique**

Cette tâche est chargée d'archiver, sur événements, les alarmes, changements d'état et commandes opérateur, et, à des périodes configurables, les mesures analogiques.

- **Tâche journaux/bilan**

Cette tâche a pour mission d'éditer, sur demande ou à des dates configurables, des journaux de bord et bilans d'exploitation. L'édition peut se faire sur imprimante ou sur écran.

- **Tâche calcul**

Cette tâche permet, à l'aide d'un langage d'accès facile (en général voisin du langage BASIC), d'effectuer des traitements mathématiques et logiques sur les variables de la base de données temps réel. Ces traitements génèrent, dans la base de données temps réel, des variables internes qui sont utilisées par les différentes tâches.

- **Tâche SPC/SQC *(Statistical Process Control/Statistical Quality Control)***

À partir de calculs statistiques sur des variables du procédé (moyenne, écart-type...) cette tâche élabore des indicateurs de rendement et de qualité qui pourront être exploités pour l'optimisation du procédé.

- **Tâche utilisateur**

La plupart des SNCC fournissent des interfaces permettant à l'utilisateur d'intégrer des fonctions complémentaires (exemple : contrôle avancé, optimisation, système expert). Ces fonctions sont assurées par des logiciels tiers ou bien réalisées par l'utilisateur, généralement en langage C.

- **Base de données de configuration et d'archives**

Cette base de données contient toutes les informations qui personnalisent la station d'interface homme-machine à un procédé particulier :

- configuration du réseau de communication ;
- contenu de la base de données temps réel ;
- liste et contenu des vues synoptiques, tendances ;

- liste et attributs des variables surveillées par la tâche alarme ;
- liste et attributs des variables à archiver ;
- contenu et format des journaux et bilans ;
- configuration de la tâche calcul ;
- configuration de la tâche SPC/SQC.

Elle contient également les fichiers des variables archivées.

Cette base de données est ouverte, grâce à des fonctions d'import/export de fichiers, des requêtes SQL ou des liens DDE (échange dynamique de données), à des applications de type SGDBR (Oracle, Access...) ou de type EXCEL.

Chapitre III

PROGRAMMATION GRAPHIQUE DES APPLICATIONS DE CONTROLE-COMMANDE

3. PROGRAMMATION GRAPHIQUE DES APPLICATION DE CONTROLE-COMMANDE

3.1 Généralités

Une application de contrôle-commande peut être définie comme un système informatique qui réalise l'acquisition de données par l'intermédiaire de capteurs et élabore des commandes envoyées au procédé physique grâce à des actionneurs. Présentes dans tous les secteurs industriels, ces applications nécessitent un développement rapide, de qualité et fiable. Habituellement réalisées à partir de langages de bas niveau (assembleurs) ou classiques (C, etc.), la programmation des systèmes informatiques destinés au pilotage de procédés physiques a été bouleversée par l'arrivée de langages graphiques plus simples, plus intuitifs et plus puissants d'un point de vue des bibliothèques disponibles.

Les ingénieurs ou techniciens, chargés de réaliser ces applications, ont généralement une formation ou une expérience basée plus sur les domaines de l'automatique ou de l'informatique industrielle que sur la programmation. De plus, ils utilisent fréquemment des méthodes graphiques de conception orientées vers les schémas blocs ou l'association de blocs fonctionnels comme le GRAFCET ou la méthode d'analyse SA-RT. Ainsi un langage de programmation graphique, fondé sur des transferts de données entre des nœuds fonctionnels, correspond parfaitement au contexte de travail des utilisateurs de ce domaine du contrôle-commande. Le langage de programmation graphique G, utilisé dans l'environnement LabVIEW™, possède toutes ces caractéristiques : expression intuitive, édition graphique, diagramme flot de données, développement de grande qualité d'un point de vue génie logiciel... De plus, ce langage permet de répondre à des applications de plus en plus larges en utilisant des bibliothèques spécifiques très nombreuses : traitement du signal, automatique, traitement statistique, logiciels de gestion des cartes d'entrées/sorties, logiciels de gestion des réseaux locaux ou industriels, etc.

3.2 Applications de contrôle-commande

3.2.1 Caractéristiques des applications de contrôle-commande

Un système informatique de contrôle-commande reçoit des informations sur l'état du procédé extérieur, traite ces données et, en fonction du résultat, évalue une décision qui agit sur cet environnement extérieur dans de sévères contraintes de temps afin d'assurer un état stable.

Les applications de contrôle-commande concernent les systèmes informatiques destinés au pilotage de procédés physiques. Ces systèmes fonctionnent souvent en ligne (on line) avec le procédé contrôlé. Dans ce contexte particulier d'interaction avec le monde extérieur, l'informatique de contrôle-commande de procédé doit satisfaire à des exigences nouvelles comme :

- le respect de contraintes temporelles adaptées au procédé piloté ;
- la gestion d'une grande diversité de dispositifs d'entrées/sorties :
- procédés continus (entrées/sorties de type continu ou analogique),
- procédés discrets (entrées/sorties de type numérique) : systèmes à événements discrets,
- procédés mixtes ;
- le respect des propriétés de la sûreté de fonctionnement, en particulier la fiabilité (continuité de service), la sécurité (garantie de la non occurrence de défaillances) ;

3.2.2 Développement des applications de contrôle-commande

L'informatique de contrôle de procédés diffère fondamentalement des systèmes informatiques classiques et conduit à la mise en œuvre de méthodes et de techniques appropriées. En particulier, la réalisation de telles applications nécessite de conduire simultanément le développement de la partie matérielle (ordinateur, cartes d'entrées/sorties, etc.) et de la partie logicielle (gestionnaire de cartes, logiciel de pilotage, interface homme-machine, etc.).

Le matériel, utilisé dans ce type d'applications, peut être très divers selon les caractéristiques opérationnelles de l'application. Ce matériel peut être une simple carte, construite autour d'un microcontrôleur, pour gérer des applications de petite taille : compteur électrique industriel, caméra vidéo, radiotéléphone, appareils électroménagers, etc. Mais il peut être constitué aussi de plusieurs microordinateurs reliés par un réseau de type bus de terrain pour des applications de grande taille ou, plus classiquement, par un simple micro-ordinateur pour toutes les autres applications.

Il est aussi important de noter que beaucoup d'applications de contrôle-commande ont été et sont encore réalisées à l'aide d'automates programmables industriels (API). Ces API présentent de nombreux avantages dans le contexte industriel, en particulier celui de la sûreté de fonctionnement avec la notion d'arrêt d'urgence et reprise après arrêt. Mais l'amélioration constante des micro-ordinateurs de type industriel, associés à des langages de programmation spécifiques, amène à les utiliser pour des applications identiques avec les multiples avantages qu'ils offrent au niveau de la conception et de la réalisation du logiciel de pilotage.

Au niveau de l'architecture logicielle d'applications de contrôle-commande, nous pouvons distinguer cinq grandes parties (Figure 3.1) :

Le traitement des données internes à l'application : analyse spectrale, loi de commande, etc. ;

- les relations avec le procédé à piloter au travers des capteurs/actionneurs et des cartes d'entrées/sorties : mesures dans le sens procédé-système de pilotage, et commandes dans le sens système de pilotage-procédé ;
- la gestion de l'interface vers l'opérateur avec la visualisation des données et l'entrée des consignes;
- la relation avec le réseau : réseau local ou bus de terrain ;
- la liaison vers une unité de stockage de masse pour la sauvegarde des données.

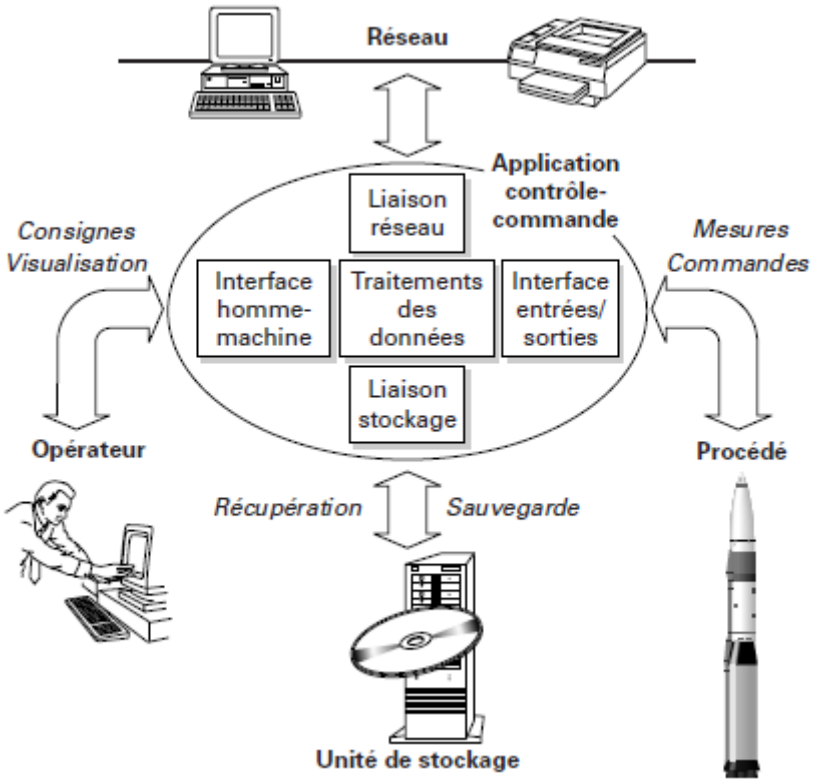


Figure 3.1 – Architecture logicielle d'une application de contrôle-commande

Ainsi, dans le cas général, le logiciel d'une application de contrôle-commande remplit successivement les trois fonctionnalités notées sur la Figure 3.2.

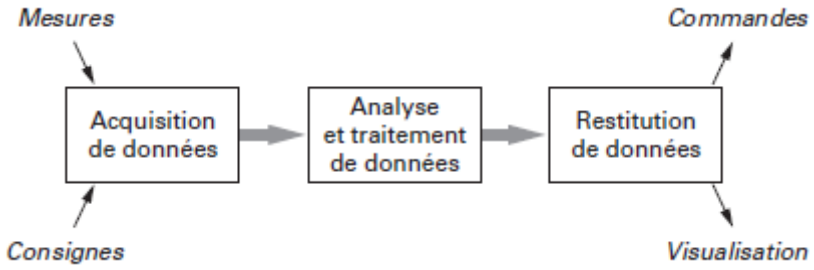


Figure 3.2 – Fonctionnalités d'une application de contrôle-commande

3.2.3 Conception des applications de contrôle-commande

Concevoir un système de contrôle-commande est un exercice difficile. En effet, la complexité et la diversité des éléments qui composent ces systèmes ne facilitent pas la description simple de leur fonctionnement. Par conséquent, une démarche méthodologique rigoureuse doit être suivie. Elle doit permettre d'obtenir une description structurelle, fonctionnelle et comportementale de l'application. Les diverses méthodes existantes, utilisées pour l'analyse et la conception des applications industrielles, peuvent être classées selon les quatre catégories suivantes :

- **Approche par les données :** ces méthodes s'intéressent principalement aux aspects structurels et informationnels du système étudié. Elles permettent d'identifier les entités qui le constituent et décrivent leurs relations ainsi que les traitements qui s'y rapportent. Nous pouvons citer le modèle entité-association et la méthode MERISE (méthodes d'études et de réalisation informatique pour les systèmes d'entreprise). Ces méthodes sont les moins

utilisées dans le domaine des applications de pilotage de procédé car elles ne proposent pas de solution concernant la description de son aspect comportemental.

- **Approche états/transitions :** ces méthodes modélisent bien l'aspect dynamique du système étudié. Elles s'articulent autour du concept des états du système, définis par un ensemble de données, qui conditionnent l'activation de transitions. Nous pouvons citer les automates à états finis, les réseaux de Pétri, et le GRAFCET.
- **Approche par les traitements :** ces méthodes sont issues du génie logiciel et modélisent les processus de génération, modification ou consommation de données. Ainsi nous avons la méthode d'analyse SA (Structured Analysis) qui permet de décomposer le processus global du système en sous-processus reliés entre eux par des échanges de données. La méthode SA-RT (Structured Analysis-Real time) ajoute à la méthode précédente l'aspect comportemental avec la notion d'activation des processus de traitement. Cette méthode d'analyse peut être couplée à une méthode de conception telle que DARTS (Design Approach for Real-Time Systems).
- **Méthodes orientées objet :** pour ne pas privilégier une approche par les données ou les traitements, ces méthodes regroupent données et traitements au sein d'un même module, appelé objet. Cet objet est une entité autonome munie d'une structure de données et de comportements définis par l'exécution de méthodes. Nous pouvons citer la méthode HOOD (Hierarchical Object Oriented Design) qui est très utilisée dans les développements d'applications spatiales européennes.

3.2.4 Programmation des applications de contrôle-commande

Comme pour tous les systèmes informatiques, le choix du langage de programmation va dépendre de la taille et de la complexité de l'application mais aussi des matériels et capacités internes à l'entreprise. Étant donné les fonctionnalités requises par ce type d'applications, le langage doit supporter les différents services suivants :

- Acquisition de données :
  - Contrôle de cartes d'entrées/sorties analogiques ou numériques,
  - Contrôle d'instruments par l'intermédiaire de liaisons normalisées : GPIB, RS232, etc.,
  - Liaison réseau (par exemple TCP/IP), base de données... ;
- traitement de données :
  - Traitement numérique des signaux (FFT, corrélation, filtrage, fenêtrage...),
  - Traitement statistique des signaux (moyenne, écart type, régression, lissage...),
  - Traitement d'images (extraction de contour...),
  - Elaboration de lois de commande (par exemple, régulation PID),
  - Maîtrise statistique du procédé (par exemple, analyse de Pareto) ;
- Présentation des données :
  - Interface graphique et interactive,
  - Représentation de courbes 2D et 3D, représentation d'images,
  - Stockage des données (impression, archivage...), génération de documents...,
  - Distribution sur réseau (Internet, Intranet...).

Étant donné l'ensemble de ces fonctions, il est très difficile de trouver un langage qui offre toutes ces composantes à la fois. Aussi, ce type d'applications est réalisé grâce à l'association de plusieurs

langages et/ou de bibliothèques de programmes spécialisées (traitement de signaux, traitement d'images, lois de commande, etc.) (**Figure 3.3**).

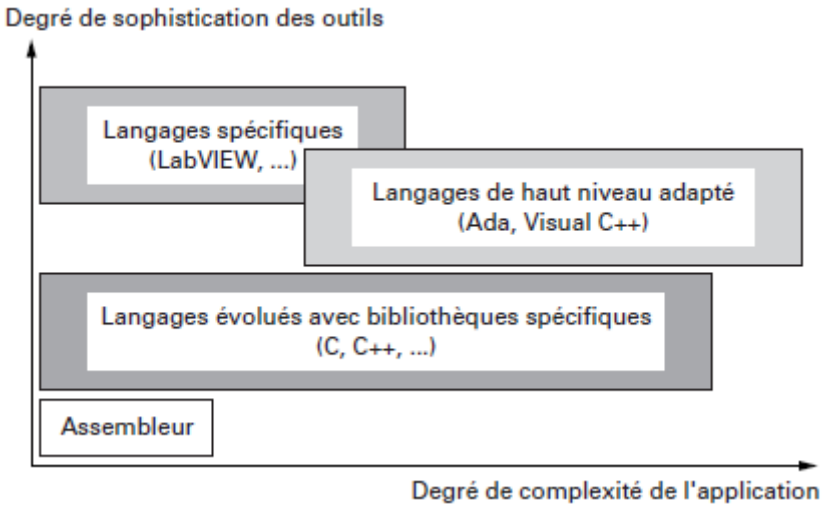


Figure 3.3 – Groupes de langages de programmation des applications de contrôle-commande

Il est très courant par exemple de réaliser les gestionnaires d'interfaces (drivers) à l'aide d'un langage de bas niveau comme un assembleur, de concevoir des interfaces homme-machine avec des langages ou bibliothèques appropriés (Visual Basic, Visual C++, Borland C++…). À l'heure actuelle, une grande majorité de ces applications sont réalisées à l'aide des langages C ou C++ qui sont associés à des bibliothèques spécifiques. D'autres langages sont également disponibles pour ces applications comme LabVIEW™ (National Instruments), MatLab™/Simulink (The Mathworks). Leur essor est lié à l'environnement de programmation offert avec un ensemble de fonctionnalités adapté à ce domaine ou à des domaines connexes comme celui de la mesure. Le langage LabVIEW, qui est détaillé dans la suite de ce chapitre, doit en particulier son développement à une programmation naturelle de type blocs fonctionnels très utilisé dans ce domaine.

3.3 Programmation graphique en contrôle-commande des procédés industriels

3.3.1 Programmation textuelle et programmation graphique

Le terme de langage de programmation graphique est employé ici pour les langages, ou environnements de développement, pour lequel le programme est décrit de manière graphique et non pour désigner des environnements de développement permettant de définir des interfaces utilisateurs (par exemple Visual Basic). La programmation à l'aide d'un langage graphique est relativement récente. Les langages de programmation textuels conventionnels impliquent un mode de conception séquentiel contraint par l'architecture même de la machine, l'exemple le plus simple étant l'assembleur puisqu'il reflète l'architecture du processeur. La possibilité de dessiner un diagramme ou le contrôle-commande d'un processus permet au concepteur d'exprimer ses idées d'une manière plus intuitive et plus naturelle. L'esprit humain perçoit et comprend les concepts compliqués plus rapidement lorsque ceux-ci sont représentés graphiquement. À cet égard, la représentation textuelle

est moins performante puisqu'elle ne permet qu'une représentation séquentielle de l'information. D'ailleurs, les efforts mis en œuvre pour rendre un texte plus rapidement compréhensible repose sur des moyens graphiques, tels que le surlignage, les italiques, les caractères gras, la couleur ou l'indentation d'un texte. Lorsque des objets peuvent être manipulés, la communication homme-machine est grandement facilitée, ce qui a conduit par exemple au développement des systèmes d'exploitation graphiques (MacOS™ ou Windows™). Les personnes ont tendance à dessiner des schémas pour expliquer le fonctionnement de mécanismes, de circuits, etc. Nous retrouvons alors la représentation naturelle de conception sous forme de « blocs fonctionnels» d'une application de contrôle-commande.
Le but de la programmation graphique est donc de faciliter la mise en œuvre d'applications informatiques.

3.3.2 Programmation *flot* de données

Un diagramme flot de données permet de décrire un programme de manière graphique, constitué des nœuds de calculs interconnectés par des arcs spécifiant le flot des données transitant entre les nœuds producteurs de données et les nœuds consommateurs de données. Le diagramme flot de données est un graphe qui peut être composé des trois éléments différents suivants (**Figure 4**).

- Des **terminaux** qui sont les liens avec l'extérieur qui représentent la production et la consommation de données ;
- Des **nœuds** qui sont les traitements à effectuer et qui sont représentés par une figure géométrique pouvant contenir une image illustrant leur fonctionnalité. Ils possèdent des connexions d'entrée et des connexions de sortie ;
- Des **arcs orientés** qui relient nœuds et terminaux et permettent d'indiquer le passage de données d'un nœud vers un autre. Un arc orienté peut se séparer en plusieurs arcs (duplication des données); par contre, des arcs orientés ne peuvent pas se regrouper (convergence interdite).

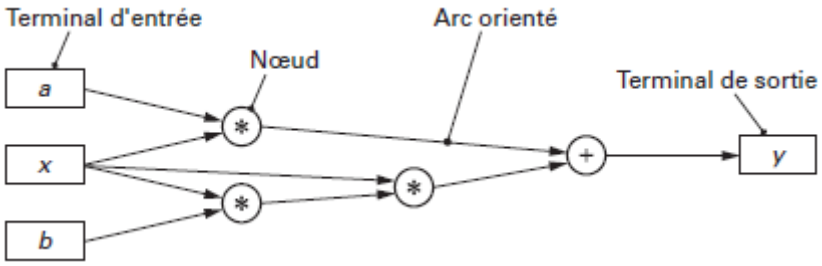


Figure 3.4 – Diagramme flot de données représentant le calcul de l'équation $y = ax + bx2$

Un diagramme flot de données peut être encapsulé afin d'être réutilisé, en tant que nœud par d'autres diagrammes flots de données. Les terminaux du diagramme deviennent alors les connexions d'entrées/sorties de ce nouveau nœud. Ce mécanisme d'encapsulation est un mécanisme d'abstraction, bien connu du génie logiciel : il permet de présenter des fonctionnalités sans décrire leur fonctionnement interne. La **Figure 3.5** montre ce mécanisme pour le diagramme flot de données de la figure 4. Formellement, cette encapsulation revient à donner un nom à un diagramme flot de données. Une fois encapsulées, les variables internes au nœud ne sont plus visibles de l'extérieur.

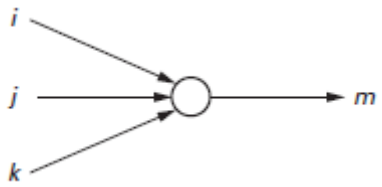


Figure 3.5 – Encapsulation du diagramme flot de données représentant le calcul de l'équation $m = ik + jk2$ explicité dans la Figure 3.4

3.3.3 Langages flot de données graphiques existants

De nombreux environnements professionnels de programmation graphique utilisent le formalisme flot de données dans les domaines de la mesure ou du contrôle-commande, comme LabVIEW (National Instruments), Visual Designer (Intelligent Instrumentation), Prograph CPX™ (Pictorius) ou bien TestPoint (Keithley).

- **LabVIEW™** (National Instruments) : c'est le premier et le plus avancé de ces environnements car il est utilisé dans le domaine de l'instrumentation, de l'acquisition de données et du traitement du signal depuis 1986. C'est également l'environnement flot de données permettant le plus large éventail de moyens de communication avec le monde extérieur : cartes d'acquisitions analogiques et numériques, bus GPIB, liaisons séries, réseaux, bus de terrains, commandes d'axes, acquisition d'images, etc. C'est donc cet environnement que nous prendrons comme exemple et que nous décrirons de façon détaillée dans la suite de ce chapitre.
- **Visual Designer** (Intelligent Instrumentation) : cet outil n'est pas réellement un langage de programmation, mais plutôt un environnement de développement d'application décrite sous la forme de blocs fonctionnels. Un ensemble de blocs fonctionnels de base est fourni, mais l'utilisateur peut et en général doit en ajouter en programmant en C++. Cet environnement est actuellement disponible sur plates-formes PC (Windows).
- **Prograph CPX™** (Pictorius) : cet environnement de développement se rapproche plus d'un langage généraliste orienté objet comme Visual C++, mais avec une programmation graphique (disponible sur plate-forme Apple, MacOS).
- **TestPoint™** (Keithley) : cet environnement de programmation, dédié à l'acquisition de données, peut être qualifié de visuel plutôt que de graphique. En effet, le graphisme est lié à l'interface utilisateur et aux bibliothèques d'objets de base, mais la création du programme est textuelle (disponible sur PC, Windows).

3.4 Langage graphique G

3.4.1 Programmation graphique flot de données

3.4.1.1 Structures de données

Le langage G est un langage fortement typé et toutes données ou structures de données ne peuvent être manipulées qu'avec des fonctions admettant ce type. Dans le langage G, on trouve les types de base scalaire : les types entiers (signés ou non, codés sur 8, 16 ou 32 bits), le type réel (codé sur 16, 32

ou 64 bits), le type booléen et le type chaîne de caractères. Il est important de noter que les éléments représentant ces données, ainsi que les liaisons issues de ces éléments, sont de forme et de couleur différente.

Le langage permet aussi de créer des structures de données plus élaborées :

- le type tableau (structure de données homogènes) : les tableaux peuvent avoir un nombre quelconque de dimensions. La représentation permet également de les différencier suivant leur type et leur dimension ;
- le type « cluster » (structure de données hétérogènes) : ce second constructeur de type est l'équivalent du « struct » en C. Les différents composants ou champs de ce groupe de données (cluster) peuvent être assemblés ou récupérés par des fonctions spécifiques.

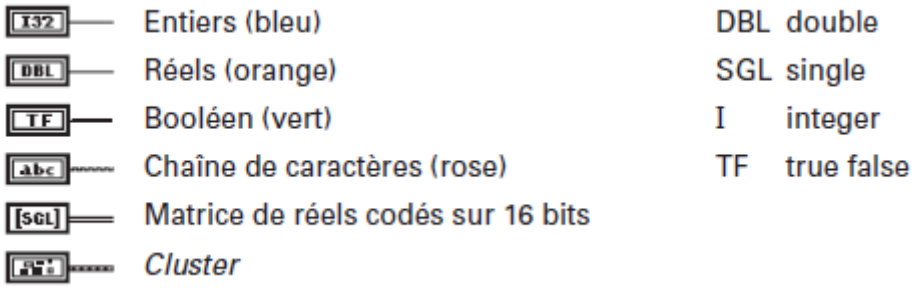


Figure 6 – Différents types de structures de données du langage G

3.4.1.2 Structures de programmation

Le langage flot de données pur a été enrichi de quatre types de structures : la séquence, deux structures d'itération (la boucle « Pour » avec un nombre d'itérations fixé et la boucle « Tant Que » avec un nombre d'itérations soumis à condition) et la structure de choix.

La **structure de « séquence »** permet de spécifier l'ordre d'exécution de flots de données. Cette structure se présente sous la forme d'un cadre et a le statut d'un nœud. Des connexions d'entrées ou de sorties à ce nœud, appelés tunnels, sont automatiquement créées lorsque des arcs arrivent ou sortent de cette structure. L'identifiant du cas représenté est indiqué en haut de la structure. Par contre, cette structure ne permet de représenter à l'écran qu'un cas à la fois, les autres parties étant cachées en arrière-plan.

Exemple : sur la figure **7**, l'utilisation de la séquence s'impose : il s'agit d'initialiser un port série puis d'envoyer une chaîne de caractères. Le premier flot de données risque de provoquer des erreurs car rien n'empêche le nœud d'écriture sur le port série de s'exécuter avant le nœud d'initialisation de ce port : l'utilisation de la séquence permet d'ordonner (de séquencer) correctement les opérations.

Les deux **structures itératives**, la boucle « Pour » et la boucle « Tant que », ont aussi le statut d'un nœud ordinaire. La boucle « Pour » permet d'exprimer la répétition (ou itération) pour un nombre de fois prédéterminé défini par une connexion d'entrée obligatoire : le nombre d'itérations à effectuer N. À l'intérieur de la boucle « Pour » se trouve un terminal d'entrée local générant l'entier indiquant l'indice d'itération de la boucle (i varie de 0 à N–1).

Exemple : l'utilisation de cette structure, présentée sur la figure **8**, concerne l'acquisition de N mesures avec un temps fixe entre chacune, défini par un nœud de temporisation. Les N données de sortie sont assemblées pour former un vecteur avec une auto indexation.

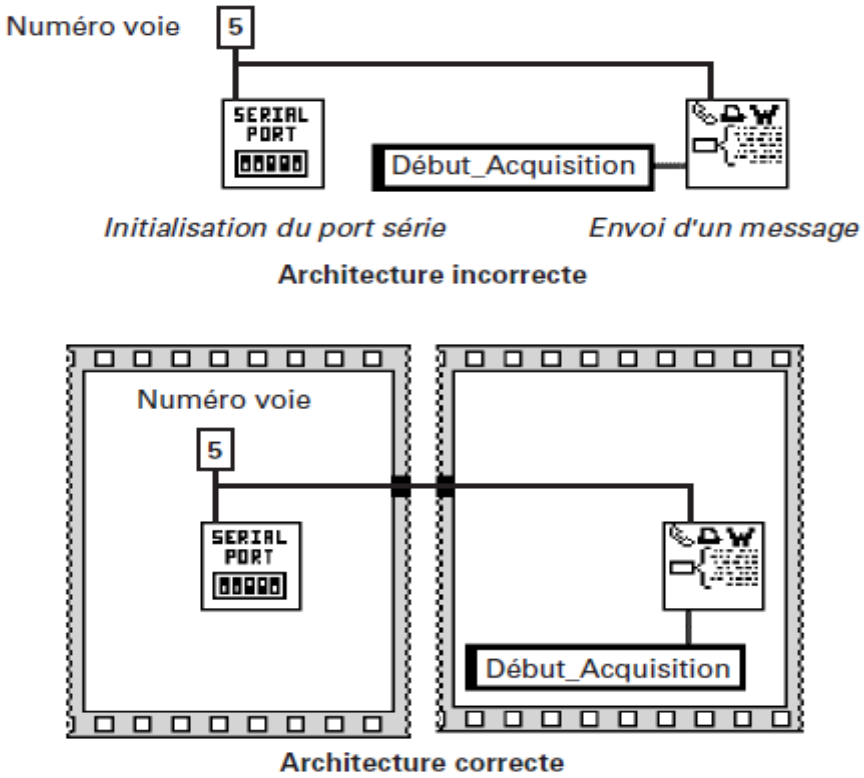


Figure 3.7 – Exemple d'utilisation de la structure de « séquence »

La boucle « **Tant Que** » permet d'exprimer la répétition pour un nombre de fois non connu à l'avance. À l'intérieur de la boucle « Tant Que » se trouve un terminal d'entrée local générant l'entier indiquant l'indice d'itération de la boucle. Un terminal de sortie de type booléen permet d'arrêter la boucle lorsque la valeur « False » lui est envoyée.

Exemple : l'utilisation précédente décrite sur la figure **8**, est reprise avec cette structure « Tant Que » en arrêtant l'acquisition pour une valeur trop grande de la mesure (figure **9**).

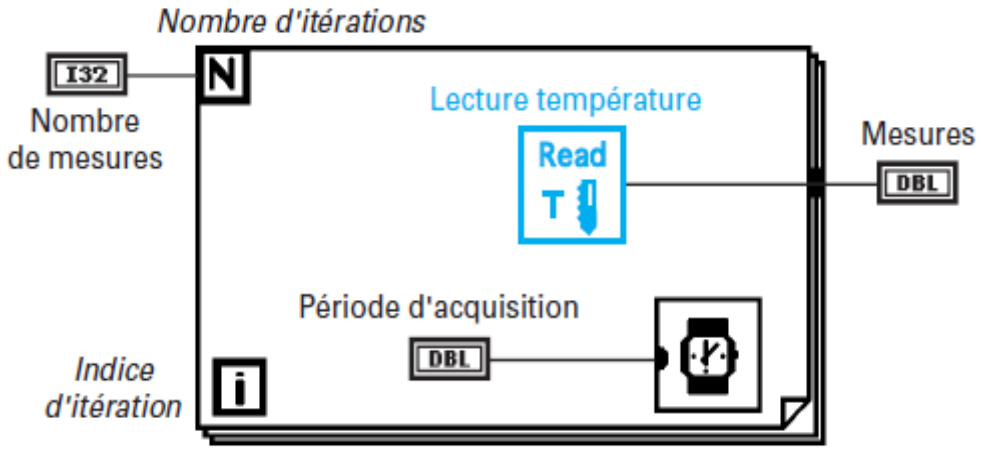


Figure 3.8 – Exemple d'utilisation de la structure itérative « Pour »

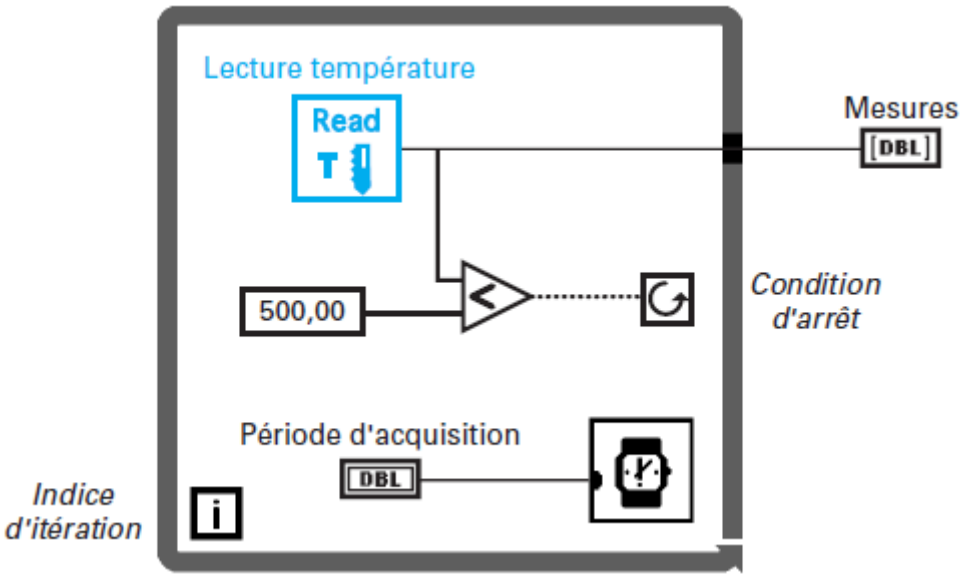


Figure 3.9 – Exemple d'utilisation de la structure itérative « Tant que »

3.4.1.3 Variables locales et globales

Pour faciliter la mise en place de la fonction de mémorisation de données, autrement qu'avec les registres à décalage des structures itératives, il a été ajouté au langage G deux types d'unités de stockage pur : les variables globales et les variables locales.

Une **variable globale** peut être considérée comme une entité possédant un point d'entrée/sortie dont la visibilité est totale. Cette variable globale va contenir un ensemble de données qui peuvent être de type différent, chacune de ces données étant accessible en écriture ou en lecture individuellement (figure **10**). Il est important de noter que d'une part cette mémorisation est un effet de bord et que d'autre part il y a possibilité d'accès concurrents et donc de conflits d'accès. Un autre inconvénient réside dans la gestion mémoire de ces variables qui peut conduire à un dépassement mémoire. Un dernier point négatif pour l'utilisation de ces variables globales est le fait qu'elles cachent le flot de données en supprimant le lien explicite et diminuent ainsi la lisibilité des diagrammes.

Les **variables locales** se présentent sous la même forme que les variables globales, mais leur visibilité est limitée au nœud contenant cette variable locale. Aussi les diagrammes flots de données appelant les nœuds, créés par duplication de ce diagramme flot de données, ne peuvent pas accéder à cette variable locale.

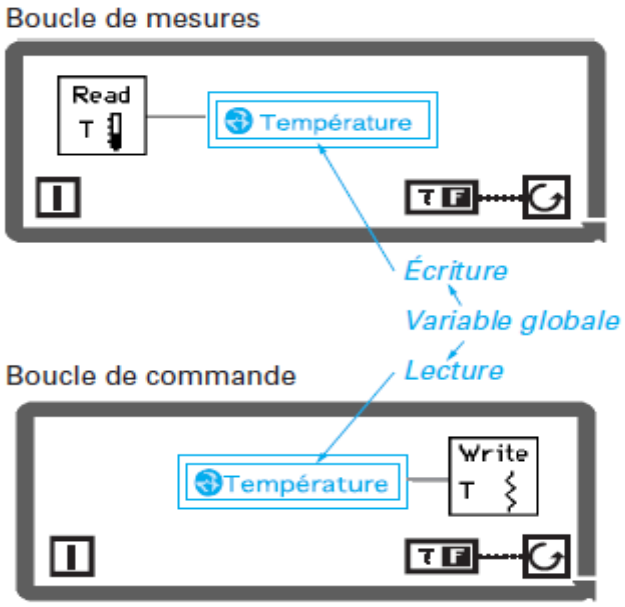


Figure 3.10 – Exemple d'utilisation d'une variable globale

3.4.1.4 Evènements

Les événements ont été introduits dans le langage G afin d'éviter de faire de l'attente active d'état de variables. Ces entités sont l'équivalent des interruptions et de leur routine de traitement dans les systèmes d'exploitation. Trois fonctions de manipulation des événements sont proposées :

- « Création d'un événement » pour créer un événement unique ;
- « Signaler un événement » pour générer un événement, c'est l'équivalent d'une interruption ;
- « Attendre un événement » pour se mettre en attente d'un événement.

Exemple : la figure 3.11 présente un exemple d'utilisation des événements pour gérer une procédure d'alarme en cas de dépassement d'une valeur de consigne.

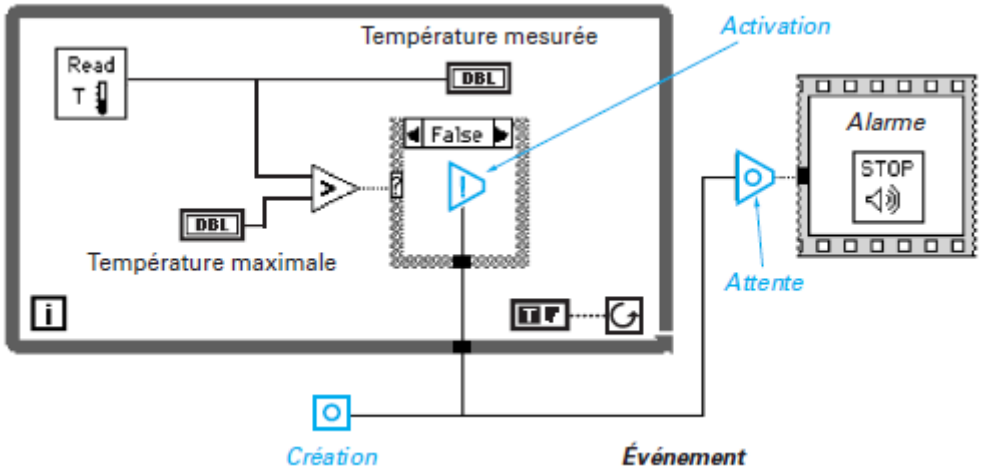


Figure 3.11 – Exemple d'utilisation d'un événement pour activer une procédure d'alarme

3.4.2 Programmation en langage G

Le langage G possède les instructions de base d'un langage de programmation permettant de traiter les différents types de données. Ainsi, nous avons des fonctions liées aux variables numériques (entiers, réels et complexes), aux variables booléennes, aux chaînes de caractères, aux tableaux.
Par exemple à la fonction « + », il est possible d'appliquer aussi bien deux entiers que deux réels et même deux vecteurs qui seront additionnés terme à terme. Nous trouvons aussi les opérateurs de test liés à ces différents types de données.
À partir des structures de contrôle, des fonctions et des opérateurs de base, il est alors possible de traduire un algorithme quelconque et d'enrichir la bibliothèque des fonctions en utilisant le mécanisme d'encapsulation. Un diagramme complet est alors réduit à un nœud qui peut être ensuite réutilisé (figure3.12).

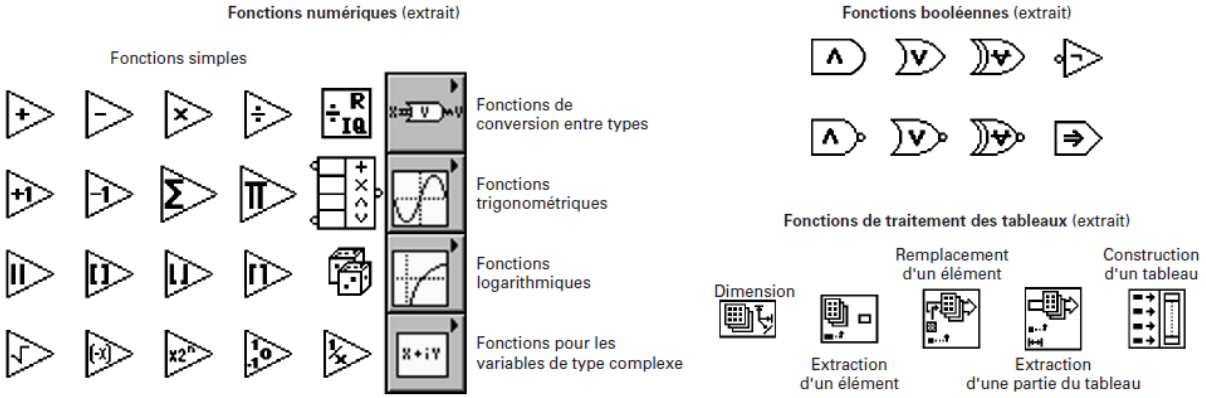


Figure 3.12 – Exemple de quelques instructions de traitement de données du langage G

Les langages de programmation graphique permettent à leurs utilisateurs (même non informaticiens) de développer facilement et rapidement des applications informatiques. Cependant, pour obtenir un logiciel de qualité, cette programmation doit se plier à certaines règles, celles du génie logiciel. On peut distinguer deux types de paramètres de la qualité du logiciel :

- les paramètres internes : modularité (finesse du découpage en modules autonomes et communicants), clarté (facilité à interpréter le code), compacité (occupation minimale d'espace de travail), logique (choix d'algorithmes rapides) ;
- les paramètres externes : validité (conformité au cahier des charges et à la spécification), fiabilité (assurer l'exactitude des résultats indépendamment des conditions et du temps), extensibilité (ajout de fonctions), efficacité (utilisation optimale du matériel), portabilité (passage du code d'une plate-forme à une autre).

Nous allons illustrer la capacité d'un langage programmation graphique, comme le langage G, à respecter les règles citées précédemment.
Ainsi, prenons l'exemple de quelques paramètres de qualité :

- la modularité : le langage G permet par sa capacité d'encapsulation des fonctions ou des données de réaliser une abstraction cohérente de l'application ;
- la clarté : dans les langages textuels, la notion de clarté est liée à l'espacement des instructions, au nombre d'instructions par lignes. Quelques règles de présentation permettent d'améliorer fortement la« lisibilité » graphique : tracé des flots de données de gauche à droite, croisement de flot de données à éviter, position des entrées à gauche et des sorties à droite, icônes avec

des graphismes clairs et si possible universels … La mise en place de commentaires associés aux liaisons ou aux nœuds est un apport considérable pour la lisibilité du programme, même si on s'écarte de la programmation purement graphique.

3.5 Environnement de développement en langage G : LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) est un environnement de développement d'applications où le langage de programmation utilisé est le langage G. Bien que tout à fait utilisable dans un grand nombre de domaines, LabVIEW est plus particulièrement destiné à l'**acquisition de données**, au **traitement du signal** et au **contrôle commande de procédé**.

LabVIEW est centré autour du principe d'instrument virtuel (Virtual Instrument ou encore VI). Un instrument de mesure analogique classique peut se décomposer en deux parties :

- la première partie, partie interne de l'instrument, est constituée des circuits électroniques de traitement et/ou d'analyse des signaux (lissage, filtrage, fenêtrage, etc.) ;
- la seconde partie (partie visible de l'instrument), appelée face avant, réalise l'affichage des résultats (par exemple, un tube cathodique d'un oscilloscope, ou des afficheurs digitaux d'un voltmètre) et permet de changer les paramètres de l'instrument (par exemple, un potentiomètre réglant l'échelle de visualisation sur l'écran de l'oscilloscope).

De la même manière, l'environnement LabVIEW va créer une application sous la forme d'un **instrument virtuel** VI de mesure composé des deux mêmes parties :

- la première partie (partie cachée ou interne) : algorithme du programme décrit sous la forme d'un diagramme flot de données en langage G ;
- la seconde partie (partie visible) est constituée de l'interface utilisateur.

Lorsque l'environnement LabVIEW est actif, nous nous trouvons en présence d'un instrument virtuel nouveau sous la forme de deux fenêtres vierges (figure **13**). Une fenêtre est dédiée à la face avant ou Panel de l'application et l'autre fenêtre va permettre d'éditer le programme en langage G (diagramme ou Diagram). Chacune de ces fenêtres est composée d'une barre de titre, d'une barre de menus déroulants et d'une barre d'exécution/édition.

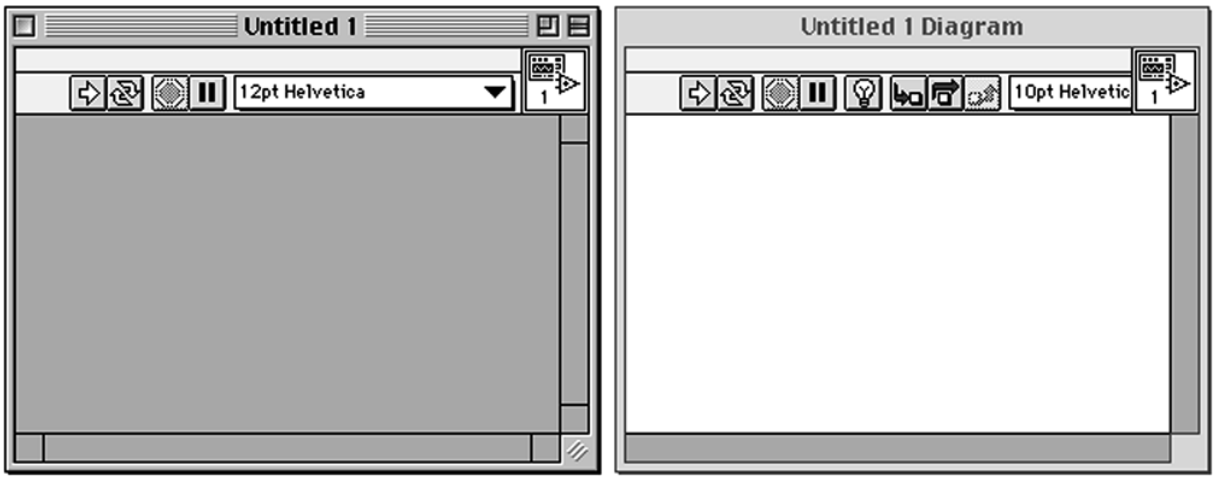


Figure 3.13 – Fenêtres de l'environnement de développement en langage G : face avant (à gauche) et programme (à droite)

La **face avant** d'un instrument virtuel est l'interface (ou moyen de communication) avec l'utilisateur. Elle symbolise la face avant de l'instrument physique. Cette fenêtre est personnalisable à souhait grâce à des objets graphiques, tels que des interrupteurs, des potentiomètres, des zones d'affichage de courbes, etc. Ces objets sont disponibles au travers d'une fenêtre, appelée **controls**, composée de sous-menus classés par type d'objet (valeur numérique, booléenne, chaîne de caractères, etc.). La figure **14** montre une face avant d'une application de contrôle-commande d'un procédé régulé en température. À chaque objet de la face avant, et suivant sa fonctionnalité (passage de paramètres ou affichage de résultat), correspond un terminal d'entrée ou un terminal de sortie dans la fenêtre de programmation (fenêtre Diagram).

Le **diagramme** d'un instrument virtuel décrit le fonctionnement interne de l'instrument virtuel. C'est dans le diagramme que l'on édite ses programmes écrits en langage G sous forme de diagrammes flot de données. Cette édition se fait en disposant des objets provenant d'une fenêtre, appelée **functions**, composée de sous-menus où sont classés par catégories les nœuds disponibles sous LabVIEW. La figure 3.15 montre le diagramme terminé correspondant à la face avant de la figure 3.14.

La création des programmes en langage G dans la fenêtre **Diagram** se fait à l'aide de différents outils de l'**environnement d'édition**. Ces outils permettent de manipuler les divers objets graphiques. Deux principaux outils servent à cette programmation en langage G : l'outil de sélection/position (flèche), utilisé pour mettre en place les divers nœuds du programme et l'outil de câblage (bobine de fil) qui sert à réaliser les liaisons entre les nœuds.

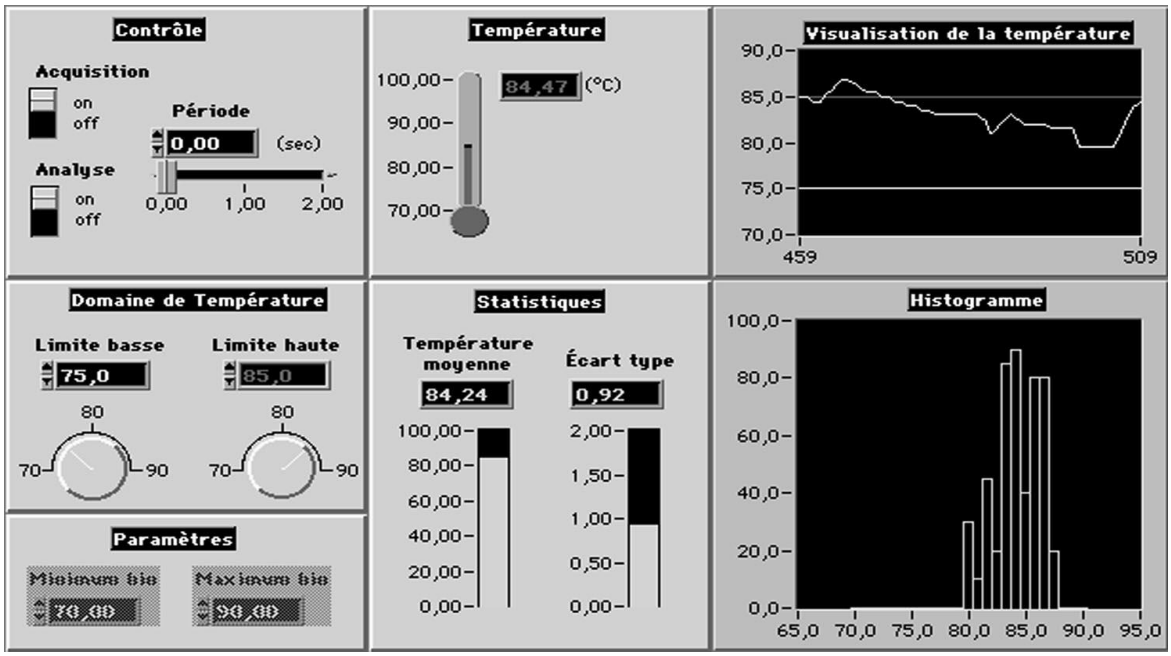


Figure 3.14 – Face avant d'une application développée avec l'environnement LabVIEW

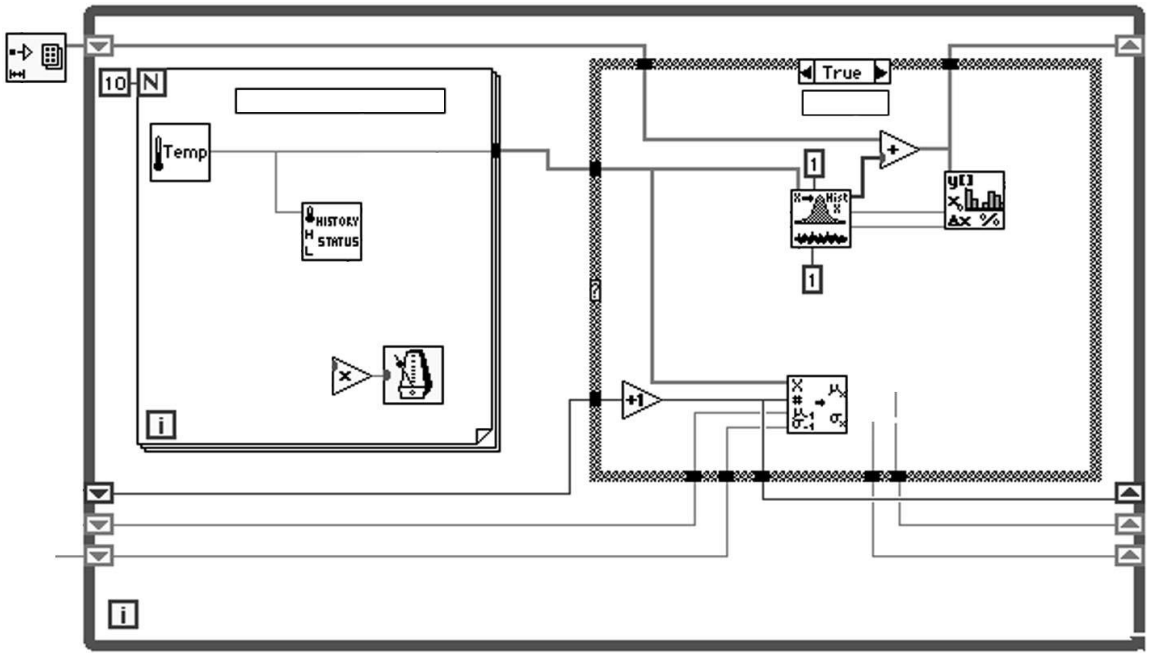


Figure 3.15 – Exemple du programme correspondant à la face avant de la figure 23 développé en langage G avec l'environnement LabVIEW

Chapitre IV

COMMUNICATION LOGICIELLE EN SUPERVISION

4. COMMUNICATION LOGICI8ELLE EN SUPERVISION

4.1 Approche objet et architecture client/serveur

Toutes les applications de contrôle-commande et de supervision utilisent des mécanismes issus de la microinformatique. Ces mécanismes qui permettent de faire communiquer les différentes applications présentes dans une installation industrielle se basent sur deux concepts :

- Approche objet
- Architecture client/serveur

4.1.1 Approche objet

L'approche objet consiste à découper un problème. Un objet structure un ensemble de données et encapsule le comportement d'un élément physique du procédé. L'objet sera créé une fois pour toutes, puis dupliqué et paramétré (instancié) autant de fois qu'il est nécessaire.

Exemple : un objet « vanne » (figure 1) peut être créé si l'on définit ses entrées/sorties E/S, puis son comportement. Il sera utilisé comme « moule », dès lors que l'on voudra inclure une vanne en programmation automate ou superviseur.

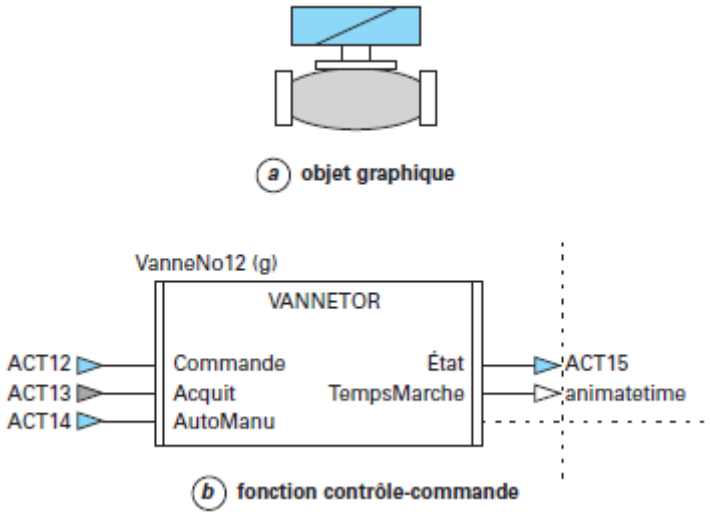


Figure 4.1 : Objet vanne

Des bibliothèques d'objets peuvent ainsi être constituées et complétées au fur et à mesure des besoins. Issue du monde informatique, cette approche permet la réutilisation du code relations d'héritage, relations d'utilisation, assemblages), ainsi que la sauvegarde du savoir-faire.
Cette approche offre aussi la perspective de mieux appréhender la complexité et la nouveauté des systèmes modulaires, réutilisables en partie ; elle est plus simple à comprendre et à maintenir. Une base de données « automatisme » sera composée essentiellement d'objets d'automatismes avec l'utilisation des protocoles de communication objet à objet : (CORBA, ActiveX). Ces objets

encapsuleront les comportements procédés fonctionnels, sécurités (machine, homme) et ergonomiques (MMI et supervision).

Exemple : l'objet graphique « vanne », issu d'une bibliothèque incluant un ensemble d'interfaces graphiques, peut avoir plusieurs représentations vis-à-vis des différentes fonctions :

- Une représentation de la fonction contrôle-commande (figure 4.1 b) : commande AutoManu, État, TempsMarche ;
- Une représentation de la fonction supervision permettant la visualisation dynamique du comportement de l'objet « vanne » (figure 4.1 a), via le logiciel de programmation ;
- Une représentation de la fonction maintenance : procédure de dépannage avec des liens vers le schéma électrique, ainsi que les références constructeurs (liens Internet éventuels pour être au goût du jour vis-à-vis du fournisseur : schémas constructeurs, références et prix).

4.1.2 Architecture client/serveur

Cette architecture est basée simplement (figure **2**) sur :

- une application **client** qui va consommer les données ;
- une application **serveur** qui va produire les données ;
- un ensemble de services établissant un canal de communication entre l'application « client » et l'application « serveur » ; ces services sont appelés **middleware** (c'est-à-dire passerelle ou couche logicielle intermédiaire).

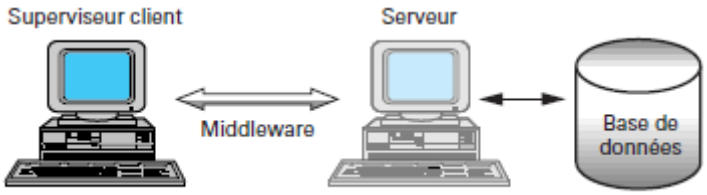


Figure 4.2 : Architecture client/serveur

Cette architecture est aujourd'hui en plein développement dans tous les milieux de l'informatique, et l'industriel voit apparaître deux formes différentes de client/serveur :

- le client/serveur de type MES, pour l'étude des valeurs à posteriori du procédé de type données de production, traçabilité, etc., dans l'architecture SGBDR;
- le client/serveur de supervision, apparu récemment, avec un serveur de communication temps réel qui diffuse ses informations vers les superviseurs clients, par le biais d'un réseau informatique classique Ethernet avec un protocole TCP/IP.

Dans les deux cas, le principe est le même. Un serveur puissant centralise les données et les met à disposition de toute application susceptible et habilitée à les traiter. La puissance et l'avancée technologique du monde de la micro-informatique peuvent fournir la puissance nécessaire à ces traitements. Ces technologies sont en pleine croissance et tirent également profit des principes de l'Intranet pour la communication TCP/IP et l'évolution du poste client en navigateur Web banalisé.

Exemple : on peut citer comme produits de développement utilisés :

- pour le client/serveur MES : Microsoft SQL Server ou Oracle avec clients Visual Basic, Delphi, Powerbuilder … ;
- pour le client/serveur supervision : PCVUE32, RSVIEW 32, INTOUCH 8, Monitor pro…

4.2 Communication DDE et DLL

4.2.1 DDE

Les DDE (Dynamic Data Exchange) sont les premiers concepts de dialogue entre applications fonctionnant sur Windows. Une communication DDE permet de rendre disponible, sur un même poste de travail, une information d'une application serveur à une application « client », en établissant un canal dédié, similaire en tous points à une conversation téléphonique sur le principe de messagerie. Pour échanger des informations, deux applications doivent engager une conversation DDE ; une application peut engager plusieurs conversations simultanément.

Une application DDE serveur met à la disposition d'applications « client » DDE sur sa base de données en temps réel. L'application « client » doit spécifier :

- le nom de l'application serveur ;
- le sujet de la conversation ;
- l'élément de la conversation.

La **figure 4.3** donne un exemple de données échangées entre un superviseur PCVUE2 et Excel. Depuis une cellule Excel de Microsoft, on peut observer la ligne de commande indiquée.

Très utilisées en supervision, les communications DDE sont très lourdes pour le système d'information et, de plus, il n'y a aucune garantie d'échange, car il n'y a pas d'accusé de réception sur l'envoi des données Les techniques DDEBlock ou AdvanceDDE ont été mises au point par des sociétés comme Wonderware pour pallier ces défauts.

Windows for Workgroups, Windows 95, Windows NT ont intégré une version du DDE améliorée pour les réseaux, appelée NetDDE ; elle permet aux applications d'établir des dialogues soit interéquipements, soit au sein d'une même machine.

4.2.2 DLL

Une DLL (Dynamic Link Library) peut se décrire comme un groupe ou bibliothèque de fonctions assemblées dans un même fichier.
Outre les fonctions qu'elle renferme avec son code, les points d'entrée et les formats de passage de paramètres, une DLL peut contenir des ressources, au sens Windows du terme. Il s'agit, par exemple, de fenêtre de dialogue, d'image au format de bitmap ou d'icône. La similitude avec une application Windows traditionnelle est d'autant plus juste que toutes les DLL de Windows ne portent pas l'extension DLL. Voici quelques exemples d'extension :

- « drv » pour le pilote d'écran ou d'imprimante ;

- « fon » pour la police de caractères ;
- « exe » pour un fichier exécutable.

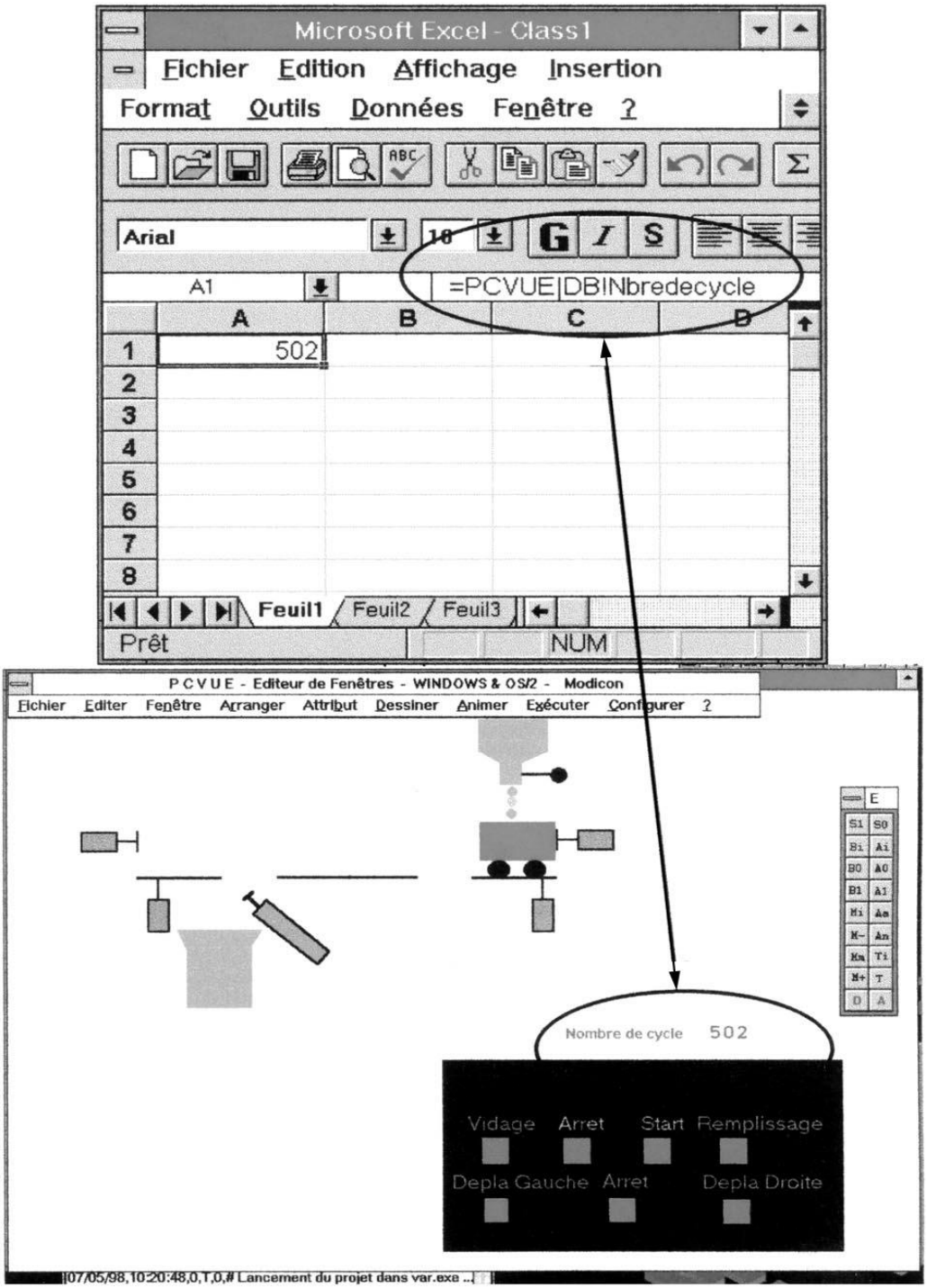


Figure 4.3 : Exemple d'échange DDE entre un superviseur PCVUE et Excel

Le **mécanisme** des DLL est le suivant : tous les langages de programmation comprennent au moins une instruction permettant de réaliser l'appel d'une procédure située dans une autre application.
Le « Call », qui fait un branchement au point d'entrée de la procédure, déroule la procédure, puis retourne à l'instruction qui suit l'appel.

L'appel d'une DLL est une extension de ce mécanisme qui autorise l'appel à une procédure située dans un autre fichier exécutable, en utilisant simplement son nom. La fonction de DLL étant terminée, le programme exe reprend et Windows libère la DLL de la mémoire.
Dans l'exemple de la **figure 4.4**, une application développée en Visual Basic peut effectuer des opérations de lecture sur les entrées/sorties (E/S) moyennant l'existence de DLL fournies et développées par le constructeur du réseau d'entrées/sorties.
L'un des avantages le plus important tient dans la réduction de la taille des exécutables. Plusieurs applications supervision (Visual Basic, Borland C++…) pourront partager les mêmes DLL, si elles invoquent des fonctions semblables (comme un ensemble de fonctions de lecture/écriture de variables). Une conséquence directe est que la mémoire de l'ordinateur ne contient pas de séquences de code redondantes, car une seule copie de la (ou les) DLL sera chargée dans la mémoire. Windows a la capacité de détecter l'utilisation d'une DLL.
Le système ne chargera en mémoire que les DLL exploitées par, au moins, un programme. Lorsque les fonctions d'une DLL ne sont plus sollicitées, Windows désactive la bibliothèque et libère ainsi la mémoire, qui devient disponible pour les autres applications.

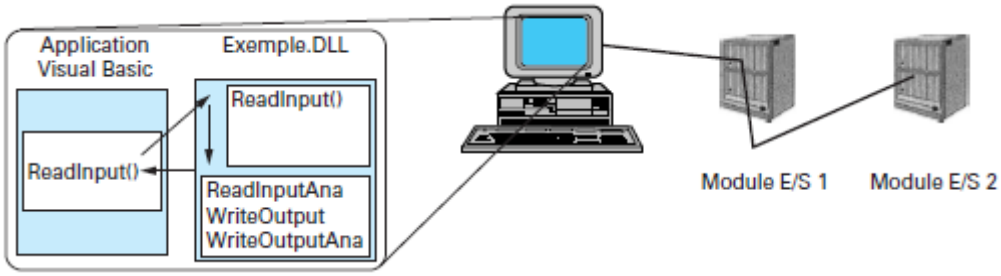


Figure 4.4 : DLL permettant d'accéder à des entrées/sorties depuis un PC

4.3 Concepts COM/DCOM et leurs mécanismes

4.3.1 OLE

L'OLE1 (Object Linking and Embedding 1) est un mécanisme permettant de créer et d'utiliser des documents composites (une feuille d'Excel dans une synoptique d'un superviseur quelconque).
La deuxième version d'OLE (OLE2) supporte mieux les documents composites que la première, dans la mesure où le document embarqué est directement modifié dans le document client (quand l'utilisateur double clique sur le document embarqué, il retrouve l'environnement de celui-ci).
Par exemple, un logiciel de supervision ayant un conteneur OLE2 complet permet d'inclure des objets graphiques d'un autre logiciel graphique. De même, si ce logiciel de supervision est serveur OLE2, les courbes de tendance et les journaux de bord peuvent être mis à la disposition d'application tierce en local ou via un réseau afin d'être publiés ou consultés à distance (**figure 4.5**).

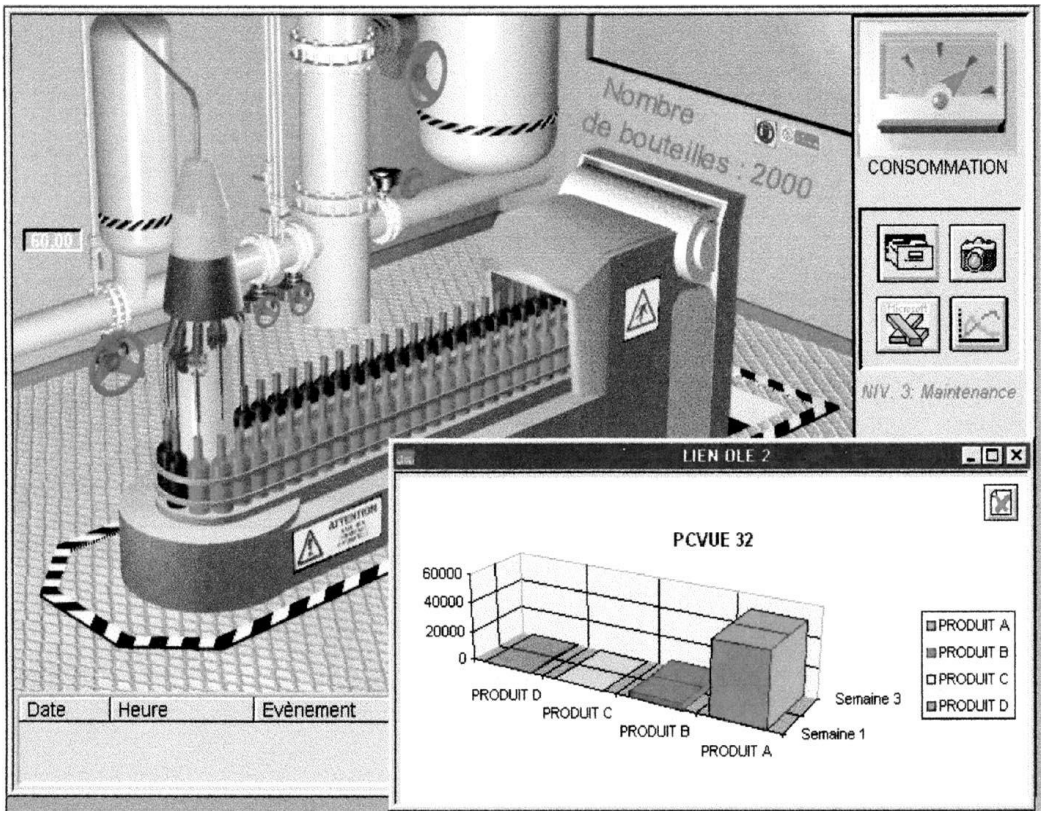


Figure 4.5 : Lien OLE2 (graphique Excel) avec un superviseur PCVUE32

4.3.2 COM/DCOM

Le potentiel utilisé par OLE est en grande partie dû à COM (Component Object Model), qui crée un modèle commun offrant une interaction entre toutes les formes de logiciels, de bibliothèques, d'applications, de logiciels systèmes, etc.

Avec COM, tout module de logiciel implémente ses services sous forme d'un ou de plusieurs objets COM. Chaque objet COM supporte une ou plusieurs interfaces, dont chacune propose un certain nombre de méthodes. Une méthode est une fonction ou une procédure qui exécute une action spécifique et qui peut être appelée par le logiciel utilisant l'objet COM (nommé client de cet objet). Les méthodes qui constituent chaque interface sont en général apparentées.
Les clients peuvent accéder aux services fournis par un objet COM en invoquant les méthodes des interfaces de l'objet, mais ils ne peuvent accéder directement aux données de l'objet.
Les objets COM peuvent être conditionnés en bibliothèques ou en fichiers exécutables, puis distribués en format binaire (sans le code source). Puisque l'on a défini ainsi un moyen d'accéder à ces objets binaires, les objets COM peuvent être écrits dans un langage et utilisés dans un autre. Et puisque les objets COM sont instanciés à la demande, lorsqu'une nouvelle version est installée sur un système, tous les clients disposent de la nouvelle version d'un objet dès la prochaine utilisation.
DCOM (Distributed COM) est une extension de COM pour le traitement distribué : un client utilise un objet qui peut être hébergé par une autre machine du réseau. Cette délocalisation est transparente pour le client de l'objet et pour l'objet lui-même.

4.3.3 OCX

Un objet OCX (Object Control Extended) est un objet COM qui possède le plus souvent une interface utilisateur, c'est-à-dire une représentation visible. L'idée de base est que ces objets peuvent être pilotés (programmés) de l'extérieur par une application « client ». Ce mécanisme de pilotage OCX s'appelle, en jargon, OLE Automation.

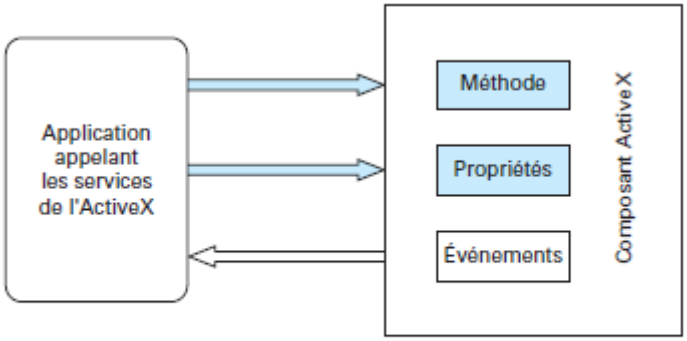
Un OCX contient des propriétés, des méthodes et permet aussi d'envoyer des événements de notification à son client.

4.3.4 Contrôle ActiveX

Les contrôles ActiveX se nommaient initialement contrôles OLE ou OCX. Microsoft a changé leur nom pour refléter plusieurs caractéristiques nouvelles qui facilitent l'utilisation avec l'Internet et le World Wide Web. La **figure 4.6** donne une structure simplifiée d'un composant.
Un contrôle ActiveX Vu-mètre à aiguille utilisable pour la supervision (**figure 4.7**) est par exemple composé :

- d'une partie graphique de représentation avec une animation pour l'aiguille, en rotation ;
- d'une feuille de propriété pour indiquer la taille, l'échelle minimal- maximal la source de communication, la variable analogique source d'entrée.

Une fois ces renseignements saisis, ce composant est utilisable dans tout container ActiveX, qui peut être un simple navigateur Web ou une feuille Visual Basic.
On4. utilise autant d'objets ActiveX que nécessaire pour réaliser facilement une supervision légère en client/serveur sur TCP/IP avec les technologies Web. La plupart des nouveaux superviseurs appliquent ces technologies.



Un composant ActiveX présente à une application, qui souhaite faire appel à ses services, des méthodes (fonctions), des propriétés (paramètres du composant) et des événements.

Figure 4.6 : Structure simplifiée d'un composant ActiveX

4.3.5 Middleware OPC

La technologie OPC (OLE for Process Control) est une extension particulière des services OLE s'appuyant sur la technologie COM.
L'utilité d'OPC est de fournir une plate-forme unique d'échange entre équipements, chaque fabricant d'appareils ou de machines ayant à sa charge le développement de cette interface. Il s'agit de normaliser les mécanismes qui permettent à un applicatif de gérer les données issues d'un autre applicatif, quels que soient les protocoles et les types d'équipements (**figure 4.8**).

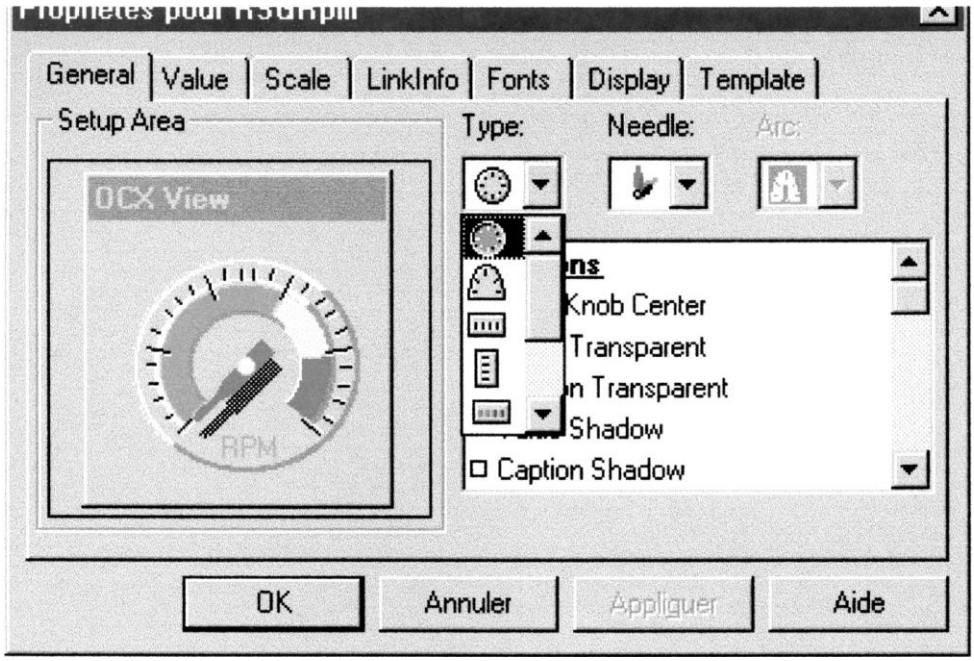


Figure 4.7 : Boîte de dialogue de paramétrage d'une jauge ActiveX

Le concept de messagerie industrielle d'où est issu OPC n'est pas neuf en soi. La première norme établie sur ce principe, permet le dialogue des équipements à travers une interface commune.

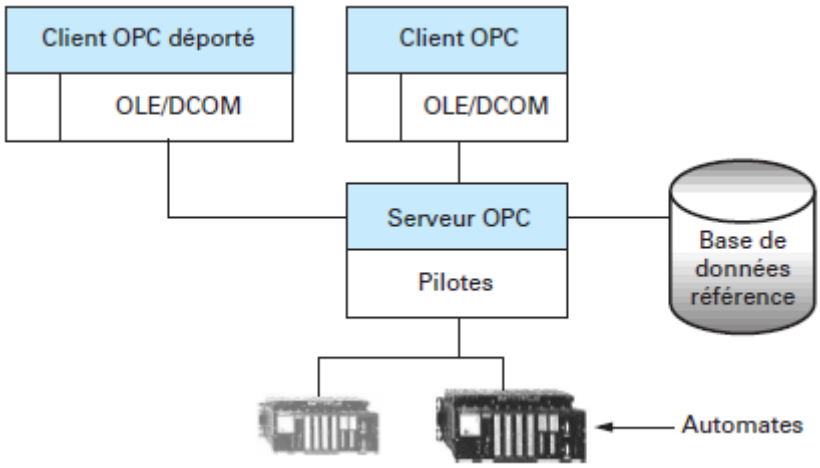


Figure 4.8 : Schéma de principe de la plate-forme OPC

La technologie DDE mise en place par Microsoft dans le but d'en faire un système de communication que l'on peut qualifier d'universel, présente un inconvénient majeur pour l'automatisme : il n'est pas sécurisé (malgré les adaptations qu'on a bien voulu faire). Une autre technologie s'est imposée : DLL est sécurisée, mais n'est pas un standard en soi ; chaque éditeur construit sa propre DLL.
La technologie OPC semble offrir les atouts de DDE et DLL : la sécurisation de l'information portée sur un standard. En effet, une information est remontée avec un attribut (drapeau) correspondant à la qualité de cette dernière.

4.4 Système de gestion de base de données relationnelle

4.4.1 Présentation

Les bases de données des applications de supervision, de contrôle-commande… comprennent des renseignements aussi divers que les libellés, les échelles physiques, les adresses des automates….
Elles sont organisées de façon plane. Les tableurs sont souvent employés pour renseigner sur ces critères, la feuille de calcul étant naturellement génératrice de ce format.
La demande aujourd'hui évolue. Par le biais de la curiosité technique, et surtout un besoin exprimé de ne pas générer une base de données spécifique pour le programme automate, puis pour la supervision, etc., les données doivent être organisées de façon à répondre au cycle de vie de l'information : de la conception de l'automatisme à l'étude des données de production.
Le SGBDR (Système de Gestion de Bases de Données Relationnelles), largement utilisé dans le cadre de l'informatique de gestion, est naturellement l'outil prédisposé à rendre ce service. Les données y sont organisées de façon rationnelle, avec un but avoué de ne pas dupliquer d'informations. La variable aura un, et un seul, libellé qui devra être utilisé par tous les programmes utilisateurs de la base. Le capteur, organe de saisie d'information, devra être qualifié et renseigné de sa tension, échelle physique, de façon unique pour l'ensemble des applications.
Le SGBDR apporte aux utilisateurs industriels les avantages qu'il possède depuis sa création :

- interdiction de doublon dans les index ;
- classement naturel de l'information avec notion de dépendance de 1 à N : une variable automate correspond, à un instant t, à une valeur procédé ; elle est forcément unique à cet instant, mais ses historiques sont multiples, les consignes également en cas de recettes séquentielles appliquées à cette variable.

En approfondissant ce cas, on se rend compte qu'il est impossible d'utiliser une feuille de calcul classique pour stocker ces informations :

- une ligne par variable suffit pour renseigner sur ses caractéristiques de base (IdVariable – identifiant variable –, libellé, unité, échelle haute, échelle basse, etc.) qui sont systématiquement utilisées, mais on ne peut y placer des consignes successives en fonction de l'avancement du procédé ;
- dix colonnes sont nécessaires pour dix valeurs à stocker et interdisent, de ce fait, la saisie d'une onzième valeur si un nouveau pas de recette est créé ; de plus, les dix colonnes sont forcément présentes pour l'ensemble des variables, alors que seules les consignes peuvent les utiliser…

L'historique d'une valeur n'est pas de l'ordre de la dizaine, mais de milliers d'enregistrements, qui doivent, par souci d'optimisation de stockage et de rapidité d'accès, ne contenir que l'information strictement nécessaire : IdVariable, valeur de mise à l'échelle (c'est-à-dire un coefficient multiplicateur) ou non, horodate de l'archive sont le minimum. Elle peut également être complétée d'un numéro de lot, d'une origine de saisie, mais, en aucun cas, de renseignements, comme le libellé de la variable, l'unité, etc., qui sont déjà donnés par ailleurs et en permanence accessibles.
On crée donc deux tables distinctes pour traiter et présenter des historiques :

- la table variable contient le nom, la valeur, l'unité, etc. ;
- la table historique contient le nom, la valeur, l'horodate de l'enregistrement.

Une relation de 1 à N est créée entre variable/tag et historique/tag qui doivent contenir la même valeur pour correspondre. À une variable peut correspondre un nombre théoriquement sans limite N d'enregistrements historiques, cela sans aucune duplication d'informations, le tag étant non pas une redondance, mais une correspondance.

4.4.2 Langage SQL

Le langage SQL (Structured Query Language) est utilisé en SGBDR pour sélectionner et filtrer l'information et présenter sous forme de résultat-tableau des informations disséminées dans la base, mais en relation les unes avec les autres de par l'architecture adoptée. Sa syntaxe est aisée à manipuler et bon nombre d'applications « client », dans une architecture client/serveur, utilisent le langage SQL comme langage d'interrogation. Présenté sous forme de phrases écrites en anglais, il est assez facile à comprendre ; il est normalisé par l'ANSI (American National Standard Institute).

Des requêtes de type SQL sont utilisées pour présenter des informations provenant de plusieurs tableaux. Il est aisé de présenter à l'utilisateur sur une même ligne, à la manière d'un tableur, une ou plusieurs lignes d'archives de la forme :

nom variable, valeur, unité, horodate.

On peut donner comme exemple de requête SQL :

```
SELECT Variables.NomVariable, Historique.HDHisto, Historique.Valeur, Variables.IdxUnite FROM
Historique INNER JOIN Variables ON Historique.IdVariable = Variables.IdVariable ORDER BY
Historique.HDHisto DESC;
```

Cette requête présente, par horodate décroissant, les noms de variables, valeurs, unités d'un historique d'acquisition.

Il faut noter que l'IdVariable, indispensable pour effectuer la relation entre la variable et ses historiques, n'est pas obligatoirement présenté, le langage SQL nous permettant d'afficher seulement les informations désirées.

L'utilisateur utilise systématiquement des résultats de SQL sans le percevoir. Le remplissage de tableaux ou de listes de choix déroulantes sur un SGBDR provient d'interrogations de forme SQL.

4.4.3 Middleware ODBC

L'arrivée des SGBDR sous Windows dans le monde industriel a naturellement profité des logiciels de liaison appelés Middleware, dont ODBC (Open Date Base Connectivity) est l'exemple le plus connu. ODBC a été initié par Microsoft et un grand nombre d'éditeurs de SGBDR dont Oracle, Sybasse, Informix, etc.

Le but ambitieux d'ODBC est de s'affranchir du format propriétaire de stockage des données pour l'application « client ». Le client adresse sa requête dans un formalisme SQL classique, sans se soucier des spécificités de tel ou tel SGBDR, la partie logicielle ODBC munie du pilote de l'éditeur se chargeant

de la traduction au format propriétaire. En théorie, il n'est pas besoin de modifier une ligne de code d'une application cliente ODBC en cas de changement de SGBDR serveur.

La **figure 4.9** donne l'exemple d'un tel pilote ODBC.
On trouve aujourd'hui une forte percée d'ODBC dans les applications « client », la légère perte de performances étant compensée par une ouverture réelle. Le tableur Excel est un des clients ODBC les plus usités. De même, un superviseur peut lire ou écrire tout ou partie de ses archives (tendances, consignations et journaux de bord) sur une base de données relationnelle (SGBD Oracle ou Access ou Ingres…) installée sur le même poste (local) ou sur un autre poste indépendant de la supervision (déporté). Une démarche similaire, de type middleware, axée « industrie » est, également, entamée avec la standardisation d'OPC.
Les avantages de ce type d'architecture sont :

- le coût réduit des composants ;
- des logiciels standards ;
- la transparence de communication ;
- des informations accessibles rapidement à tout moment et n'importe où.

Quelques points sont encore à améliorer aujourd'hui : la sécurité et la fiabilité des échanges d'informations.

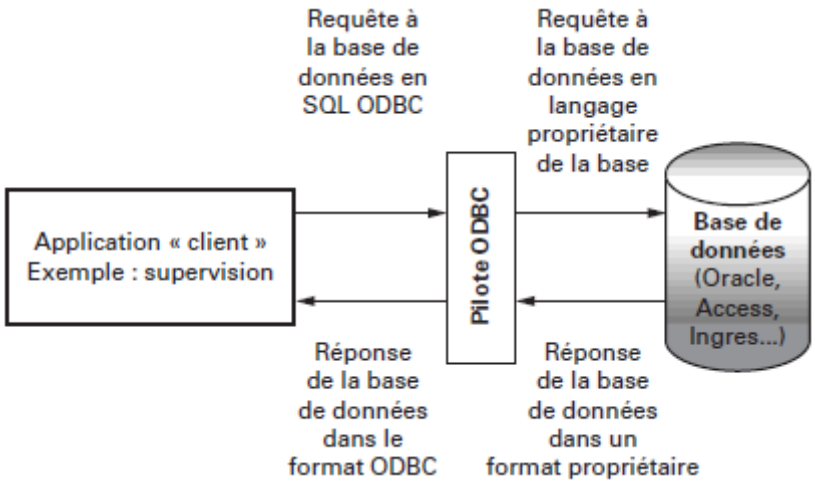


Figure 4.9 : Pilote ODBC

Bibliographie

- A. Mtibaa, *Les systèmes automatisés de production*, ENIM 2012
- J.C. Busy et D. Mérat, *Automatisme appliqué*, Edition EDICALIVRE
- F. Degoullange, R. Lemaître et D. Perrin, *Automatismes: Option Technologies industrielles*, Edition DUNOD, 1982
- Michel Bertrand, *Automates Programmables Industriel*, Techniques de l'ingénieur, S8015
- Michel Blanchard, *Comprendre maitriser et appliquer le grafcet*, Cépadues éditions, 1994
- Jean Yves Farbert, *Automatismes et automatique*, Editions Ellipses, 2005
- *Rapport technique - Programmer avec Step7 V5.4*, Siemens Edition 2006
- *Rapport technique - SIMATIC : Outils d'ingénierie S7-PLCSIM V5.4 incl. SP3*, Siemens Edition 2009
- *Rapport technique - SIMATIC WinCC flexible Flexibilité dans toutes les applications IHM du Micro Panel au PC*, Siemens Edition 2010.
- Guy Gauthier et Ilian Bonev, *Manuel de formation: L'ingénieur en production automatisée*, École de technologie supérieure - Département de génie de la production automatisée, Montréal, Session Hiver 2007