Développement d'une IHM basée sur le standard OPC

Serge Bouter, Rachid R. Malti

DEVELOPPEMENT D'UNE IHM BASEE SUR LE STANDARD OPC

Serge BOUTER[1] et Rachid MALTI[1,2]
{serge.bouter, rachid.malti}@iut.u-bordeaux1.fr
[1]IUT Bordeaux 1, 15 rue Naudet, CS 10207, 33175 GRADIGNAN Cedex
[2]IMS UMR 5218 CNRS, Département LAPS – Université Bordeaux I
351, cours de la Libération – F 33405 Talence cedex – France

RESUME L'article rapporte une expérience d'enseignement sur le développement d'une application de supervision, réalisée dans le cadre de la Licence Systèmes Automatisés et Réseaux Industriels de l'IUT Bordeaux 1. L'enseignement des systèmes automatisés au département repose sur un ensemble de parties opératives pilotées via le réseau Ethernet ou des bus de terrains, ce qui permet aux étudiants d'aborder l'essentiel de cette discipline. Ce projet de supervision d'une maquette, le « chariot élévateur » utilise le protocole OPC et met en oeuvre au niveau de la programmation l'interface des composants « ActiveX ». Sur ce dernier point, des connaissances sur la programmation événementielle sont essentielles. Le projet est conçu comme une séance de TP (4h) qui fait partie d'un module pratique concluant leur formation théorique.

Mots clés : supervision, réseaux, serveur OPC, programmation événementielle, ActiveX

1 PRÉSENTATION DU CONTEXTE

1.1Enseignement des systèmes automatisés

Les systèmes automatisés s'appuient de plus en plus sur des dispositifs de commandes réparties connectées à divers types de réseaux ou de bus de terrain. Ainsi l'enseignement dans le cadre de la licence Systèmes automatisés et Réseaux Industriels (SARI) permet aux étudiants d'aborder tous les aspects logiciels et matériels de la discipline.

Aussi la licence SARI, inclut dans son programme, les systèmes de supervision.

1.2 Objectifs pédagogiques et pré-requis

Le développement d'une application de supervision conduit:

- à montrer l'interactivité entre le monde des automates programmables industriels et celui des micro-ordinateurs de bureau,
- à mettre en oeuvre « rapidement » une interface homme/machine
- à utiliser les méthodes d'accès au serveur OPC, offertes par l'interface des composants ActiveX.
- à appliquer la programmation évènementielle

Ce projet apparaît en fin de formation et nécessite des pré-requis importants: algorithmique, programmation orienté objet, programmation évènementielle. Il est précédé d'un cours qui introduit des notions propres à la supervision : interface homme/machine, lien Net DDE, objets COM, DCOM et serveur OPC. Ce cours doit montrer comment sont partagés les données dans un environnement de commande et de supervision distribués.

1.3 Problème étudié

Un système de commande en réseau est un ensemble de composants matériels et logiciels en interaction avec un processus que l'on souhaite commander.

La partie supervision doit gérer la coordination entre les divers éléments de commande répartie. Donc il est nécessaire d'assurer l'interopérabilité entre des dispositifs ayant des technologies matérielles et logicielles différentes. Dans le cadre de ce projet, c'est le protocole OPC qui permet cette interopérabilité et c'est ce qu'illustre la figure 1.

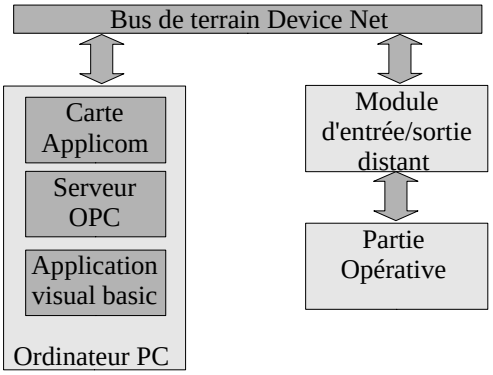


fig 1 : interopérabilité

1.4 Description du poste de travail

Le sujet de TP est construit autour de la partie opérative « chariot élévateur », qui est montrée sur la figure 2.

Les capteurs et actionneurs de la partie opérative sont reliés à un module d'entrées/sorties « distant » connecté au bus de terrain DeviceNet (voir figure 1). Le poste de travail comporte:

-la partie opérative

-un PC équipé d'une carte ApplicomIO (WoodHead) intégrant un contrôleur de bus DeviceNet

-les logiciels de développements et de configurations nécessaires au sujet

fig 2 : maquette « chariot élévateur »

Le développement de l'IHM s'appuie sur le l'environnement de développement intégré Visual Basic .NET.

Le dialogue entre la carte ApplicomIO est réalisé à travers un serveur OPC. Ce dernier est configuré à partir du logiciel « ActiveX configuration » d'ApplicomIO.

Le protocole OPC s'appuie sur une architecture Client/Serveur . L'accès au serveur est réalisé par le contrôle ActiveX « AppOcxClientcontrol » qui offre une interface de gestion des objets distribués, basée sur le modèle COM/DCOM, et ainsi permet d'accéder au serveur OPC.

Quant à la carte ApplicomIO, elle assure l'échange des données selon des modes définis par le protocole DeviceNet. Ces échanges sont configurées avec le logiciel « console ApplicomIO » qui permet de configurer le bus de terrain DeviceNet.

1.5 Organisation de la séance de travaux pratiques

Ce module pratique s'étend sur une séance de quatre heures. Les étudiants travaillent en binôme et sont déjà familiers avec la maquette sur laquelle ils ont déjà travaillé, sur un sujet traitant de la supervision à travers le protocole NetDDE. Ainsi, ils connaissent déjà les logiciels associés à la carte Applicom.

2 PRÉSENTATION DE LA NORME OPC-DA

L'objectif des différentes normes OPC est de standardiser les échanges de flux entre équipements communicants sur des réseaux locaux industriels hétérogènes. Le développement d'Interface Homme-Machine (IHM), la gestion des évènements et alarmes, et le stockage de données en provenance du réseau se trouvent ainsi considérablement facilités.

Les standards OPC se déclinent sous plusieurs versions:

- OPC Data Access (DA), la norme la plus répandue en industrie, permet de développer des IHM et de superviser une installation industrielle.
- OPC Alarm and Events (AE) permet de gérer plus facilement les alarmes et les évènements intervenant sur un réseau industriel.
- OPC Historical Data Access (HDA) permet de sauvegarder l'historique des données.

Dans cette présentation nous nous focaliserons essentiellement sur la norme OPC-DA, utilisée par les étudiants en TP. Selon les spécificités de cette norme chaque item, représentant une donnée, est constitué de trois champs :

- *Value* : contient la valeur de la donnée. Son format (booléen, entier, réel,...) dépend de l'équipement supervisé.
- *Time-stamp* : indique l'instant de transmission de la donnée de l'équipement réseau vers le serveur OPC.
- *Quality* : indique le degré de validité des données. Elle peut être « bonne, douteuse, ... ».

Un serveur OPC est structuré sous une forme hiérarchique contenant une racine, des répertoires (ou groupes) pouvant contenir d'autres répertoires, et des items, comme le montre la figure 3.

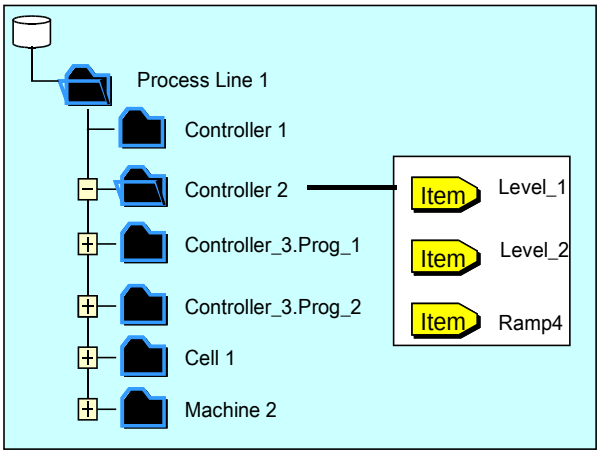


fig 3 : Organisation des objets sous OPC

Ainsi chaque item peut être référencé de deux manières distinctes :

- En utilisant le chemin d'accès complet à partir de la racine,
- En utilisant un identifiant unique connu sous le nom de « *Fully Qualified ItemID* ».

2.1 Modèles de lecture et d'écriture sous OPC

Selon les spécificités de la norme OPC-DA, les interfaces (ou méthodes) de lecture et d'écriture s'appliquent sur un groupe d'items et non pas sur un item unique. Trois modes de communication existent.

2.1.1 Mode synchrone

Il s'agit là d'un mode d'interrogation classique (polling) où les fonctions de lecture et d'écriture sont bloquantes, en attente d'une réponse en provenance du serveur OPC (figure 4).

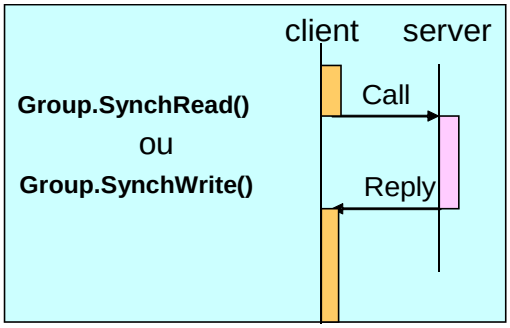


fig 4 : Lecture ou écriture synchrone

2.1.2 Mode asynchrone

Il s'agit là d'un mode d'interrogation où les fonctions de lecture et d'écriture ne sont pas blocantes. Ainsi, le client, après avoir émis un ordre de lecture ou d'écriture, continue l'exécution de son programme. Dès que le serveur OPC exécute l'ordre demandé en informe le client par la notification d'un évènement (figure 5).

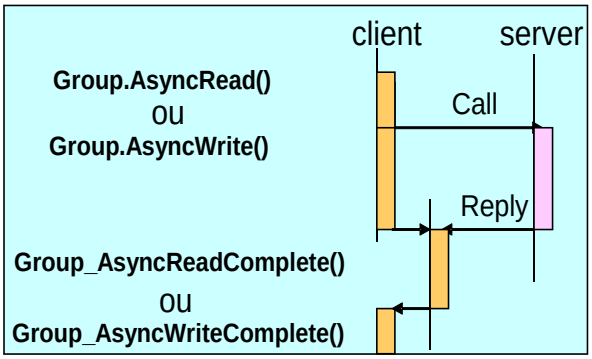


fig 5: Lecture ou écriture asynchrone

2.1.3 Mode notification (valable uniquement pour la lecture)

Ce mode permet de notifier au client tout changement de valeur pouvant intervenir sur les items. Il est par conséquent nettement plus avantageux que le mode synchrone ou asynchrone, car il évite la scrutation régulière d'un ou plusieurs items.

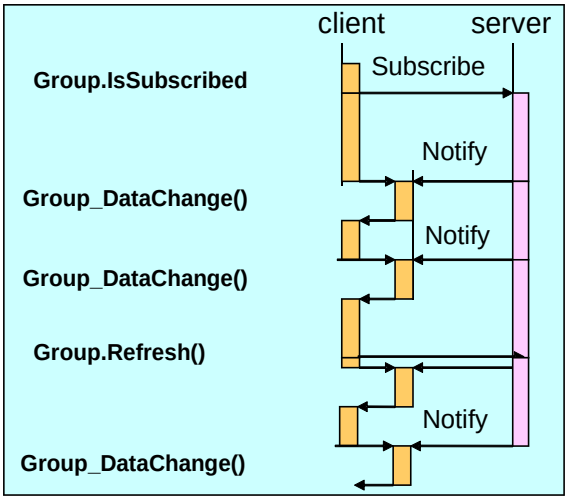


fig 6: Notification de changement d'état

Les notifications sur des variables analogiques ont lieu, à des pas d'échantillonnage définis par la propriété **Refreshrate** du groupe d'items, si la différence depuis la dernière mesure excède la zone morte définie par la propriété **Deadband** du groupe d'items.

3 DÉVELOPPEMENT DE L'INTERFACE HOMME/MACHINE ET DE LA SUPERVISION

3.1Objectifs

Les étudiants doivent développer une application qui comporte quatre parties

- Interface Homme Machine (IHM)
- gestion des modes de fonctionnement
- Cycle automatique
- Fonctionnement manuel

La série d'exercices présentée aux étudiants conduit au développement de l'application de supervision.

Dans un premier temps, les étudiants établissent la liste des capteurs et des actionneurs équipant la maquette et configurent le serveur OPC en conséquence.

Les exercices de programmation sous Visual Basic, sont proposés dans l'ordre suivant:

- fonctionnement manuel: il s'agit de commander les différents actionneurs de la maquette à partir de «clics sur des boutons »
- visualisation de l'état des capteurs à partir de « voyants »: les étudiants mettent en oeuvre une fonction de notification « Event_NewValue » déclenchée par le changement de valeur d'une variable au niveau du serveur OPC. A cette étape les étudiants ont abordées l'essentiel des outils donnant accès au serveur OPC.
- codage d'un GRAFCET « cycle automatique »
- gestion des modes de fonctionnement

L'IHM est réalisé au fur et à mesure de la mise en place, sous Visual Basic, de « boutons » et

« voyants ». L'organisationde l'IHM est laissé à l'initiative des étudiants. Mais ces derniers doivent veiller à ce que la mise en forme de l'IHM soit cohérente.

3.2 Configuration du serveur OPC

Dans cette phase, l'outil de configuration liste l'ensemble des données provenant du module d'entrées/sorties distant récupérées par la carte Applicom. Il s'agit de déclarer un ensemble des variables au niveau du serveur OPC(voir figure 7) et de les associer aux entrées/sorties du module distant.

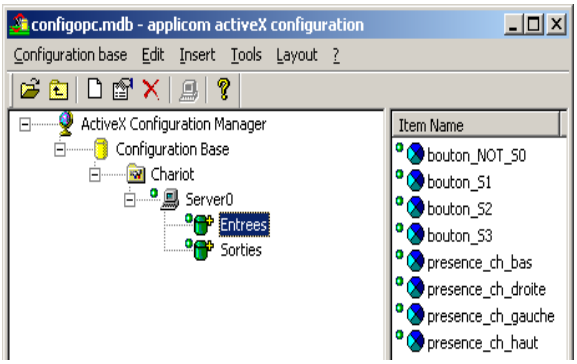


fig 7 : configuration du serveur OPC

Dans notre cas les variables sont du type booléenne et représentent l'état des capteurs et actionneurs de la partie opérative. Les variables sont séparées en deux groupes, les entrées et les sorties. Chacune des variables est repérée par une étiquette appelée « Item ». Cette étiquette est formatée sous la forme d'une chaîne de caractère et doit être représentative de la fonction et de la position du capteur ou de l'actionneur qu'elle représente.

Exemple: « Server0.Entrees.presence_ch_bas »

3.3 Phase d'initialisation du serveur OPC

Durant cette phase, l'application « active » le serveur OPC; Il charge en mémoire le programme de gestion des accès au serveur OPC. Il est aussi possible de « désactiver » le serveur OPC.

Puis il s'agit de récupérer toutes les informations concernant la configuration du serveur : nom de la configuration, groupes et « Items » associés aux variables déclarées au niveau du serveur.

Dans l'exemple ci-dessous l'objet « client » est l'instanciation du composant ActiveX. Ces lignes de code permettent d'activer le serveur et récupérer des informations nécessaires aux méthodes du composant ActiveX.

```
Retour = Client.ActiveConfig
RefServer = Client.GetServerRef("Server0")
RefGSorties = Client.GetGroupRef(RefServer, "Sorties")
Name =  "ch_droite_vers_droite"
RefItemS_d_d = Client.GetItemRef(RefGSorties,Name)
```

3.4 Fonctionnement manuel

Le programme gérant le fonctionnement manuel s'appuie essentiellement sur des boutons et les fonctions de lecture et d'écriture offertes par le composant ActiveX.

3.4.1 Fonction d'écriture

La fonction membre « Write » de l'objet « client »permet de modifier la valeur des variables déclarées au niveau du serveur OPC. Cette modification est répercutée sur la maquette, et permet la marche ou l'arrêt des moteurs..

```
Dim tabStatus As Object
Dim retour As Long
retour = Client.Write(1, RefItemS_d_d, valeur, tabStatus)
If retour = 0 Then
        status.Text = "Write ok"
Else
        status.Text = "Write bad"
End If
```

3.5 Fonction de lecture

La fonction membre « Read » de l'objet « client » permet de lire la valeur des variables déclarées au niveau du serveur OPC. Cette solution n'a pas été envisagée. La prise en compte du changement de valeur d'une variable au niveau du serveur OPC est réalisée par une fonction de notification déclenchée sur évènement. .

Les états des capteurs associés à la maquette sont intégrés dans des variables « réseau ». Ainsi le changement d'état d'un capteur provoque une modification de la valeur des variables « réseau » et par conséquent des données déclarées au niveau du serveur

Le composant ActiveX utilisé offre une fonction de notification « sensible » au changement de valeur des variables déclarées au niveau du serveur OPC.

Ainsi, dès que une variable au niveau du serveur OPC est modifiée, la fonction de notification « *Client_EventNewValue* » est exécutée. Dans un premier temps il nécessaire d'effectuer une séquence identifiant la variable ayant subi une modification. Cette séquence se compose d'une boucle « «FOR » et d'une structure à choix multiple.

```
Private Overloads Sub Client_EventNewValue(...) ...
        ...
        For i = 1 To e.nbItems
            status.Text = e.tabItemName(i)
            Select Case e.tabItemName(i)
                ...
                Case "Server0.Sorties.M0"
                    flagM0 = e.value(i)
                    If flagM0 Then
                       M0.BackColor = Color.Red
                    Else
                       M0.BackColor = Color.White
                    End If
                Case "Server0.Sorties.M1"
                    flagM1 = e.value(i)
                    If flagM1 Then
                       M1.BackColor = Color.Red
                    Else
                       M1.BackColor = Color.White
                    End If
                       ...
                End Select
                       ..
            End For
```

Cette partie de programme positionne la couleur(rouge/Blanc) du voyant associé à la variable sur lequel un changement de valeur à été détecté.

A cette étape du développement, les étudiants obtiennent déjà une IHM telle que celle qui est représentée sur la figure 8. L'enseignant peut alors évaluer la cohérence de l'IHM et conseiller les étudiants sur leurs choix.

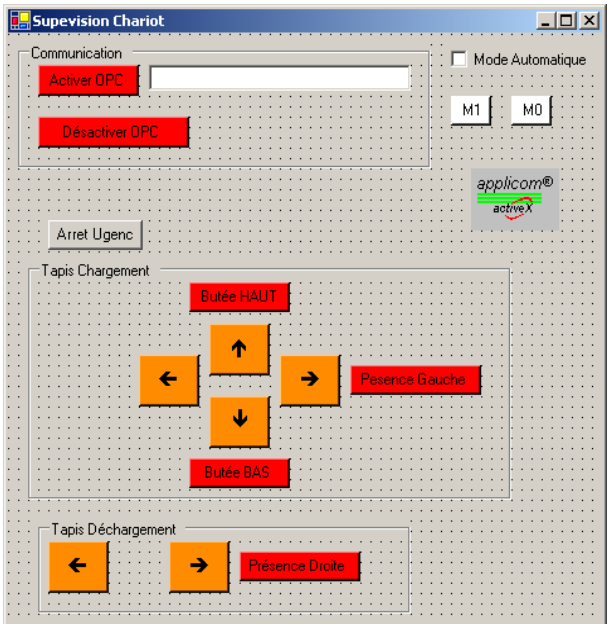


fig 8 : IHM obtenu sous Visual Basic

3.6 Déclenchement de la fonction de notification « *Client_EventNewValue* »

Les outils de débogage de Visual Basic permettent aux étudiants d'observer ce qui est récupéré lorsque la fonction de notification « *Client_EventNew-Value* » est déclenchée (voir figure 9)

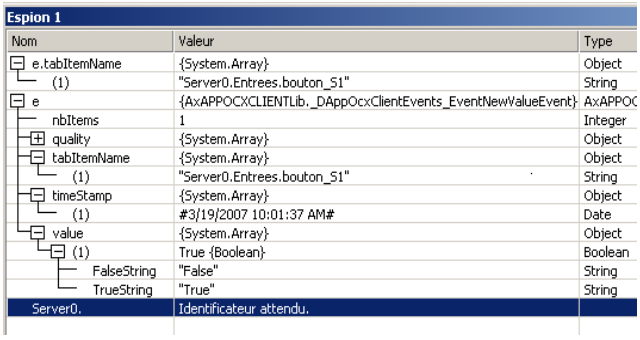


fig 9 : débogage sous Visual Basic

Ainsi les étudiants peuvent repérer la donnée membre à tester pour obtenir l' « Item » de la variable modifiée. De plus, ils peuvent constater qu'un ensemble d'informations accompagne la variable et que ceci est conforme à la norme OPC.

3.7 Cycle automatique

La partie de programme « cycle automatique » est intégrée à la fonction de notification « *Client_EventNewValue* » *et* peut être organisée en plusieurs phases sucessives

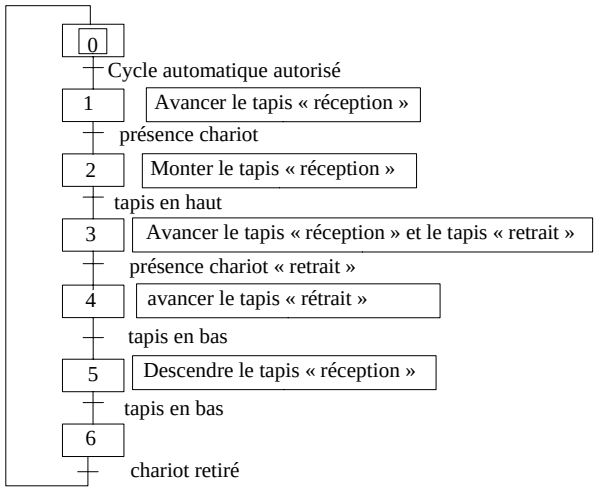
-test de la variable ayant subi une modification.

-traitement du grafcet

-l'élaboration des commandes

Le test de la variable modifiée est déjà traité dans la partie permettant de visualiser l'état des capteurs.

3.7.1 Grafcet du cycle automatique



Dans le sujet, le Grafcet à établir est relativement simple. Il ne s'agit pas d'évaluer les étudiants sur cette partie.

3.7.2 Codage du Grafcet

réceptivités

```
....
receptivite1 = flagBG
receptivite2 = flagBH
receptivite3 = Not flagGH
....
```

Les réceptivités sont un ensemble de conditions qui est associé à une transition entre deux étapes. Dans cette partie de code, la valeur de la réceptivité est obtenu à partir des variables testées précédemment.

Activation et Désactivation des étapes

```
....
If receptivite0 And etape0 Then
                etape1 = 1
                 etape0 = 0
Endif
If receptivite1 And etape1 Then
                etape2 = 1
                etape1 = 0
Endif
...
```

Cette phase teste les transitons. Si une transition est validée (étapes précédentes actives et réceptivité vrai), elle est franchie (étapes suivantes activées et étapes précédentes désactivées).

Élaboration des commandes

```
....
        If etape1 Then ecrire(RefItemGD, 1)

        If etape2 Then
           ecrire(RefItemGD, 0)
           ecrire(RefItemGH, 1)
        End If

        If etape3 Then
           ecrire(RefItemGH, 0)
           ecrire(RefItemDD, 1)
           ecrire(RefItemGD, 1)
        End If
```

.....

L'état du Grafcet obtenu, il s'agit de positionner les actions en fonction des étapes activées

4 CONCLUSIONS

4.1 Enseignement de la Supervision en licence SARI

Ce projet permet aux étudiants de mettre en oeuvre rapidement un serveur OPC et de montrer l'intérêt de ce protocole dans un environnement hétérogènes (automates programmables industriels, modules d'entrées/sorties distant, ordinateur de bureau...). De plus, les étudiants ont pu réaliser « très rapidement » un client OPC et une Interface Homme/Machine grâce à un ensemble d'outils tels que les composants « Activex » et l'environnement de développement rapide « Visual Basic ».

4.2 Perspectives

Le standard OPC-DA est dépendant des objets COM et DCOM de Microsoft et ainsi du système d'exploitation Windows. De plus, il est nécessaire d'ouvrir des ports particuliers correspondant aux objets DCOM pour une supervision via Internet.

Le standard OPC XML-DA, publié en 2003, est le premier standard OPC à s'affranchir des objets COM et DCOM de Microsoft. Les fonctionnalités sont les mêmes mais avec des Services Web basés sur le formatage XML des fichiers traités. Ainsi, les nouvelles interfaces hommes-machines peuvent s'exécuter sur n'importe quel navigateur Web sans avoir à déployer les applications développées sur des postes clients. De plus, cette nouvelle norme s'affranchit complètement du système d'exploitation et ne nécessite pas l'ouverture de port Ethernet supplémentaires car l'échange de données se fait via le port 80 (port http).

5 BIBLIOGRAPHIE

[1] Site Web de la fondation OPC : http://www.opcfoundation.org/

[2] Applicom Communication Active X Control. 2001.

[3] ApplicomIO Documentation 2.2. 2002.