Unity fait entrer les systèmes dans vos PC

STÉPHANE PITAULT [1]

Pour former les étudiants du BTS Maintenance des systèmes à la programmation d'un système automatisé plusieurs méthodes sont utilisables. Elles dépendent du degré d'acquisition de cette compétence par les étudiants. Avec Unity, ils peuvent débuter en toute confiance.

mots-clés
travaux pratiques, simulation

En BTS MS, afin de vérifier la compétence C24 « Identifier et caractériser la chaîne d'information associée au savoir S7.3 : programmation des systèmes de traitement », il est nécessaire de mettre nos étudiants en phase d'apprentissage sur la programmation des systèmes. Mais les programmes qu'ils imaginent sont rarement valides et opérationnels du premier coup. Il est donc délicat de les tester directement sur les systèmes réels, les conséquences d'une erreur pouvant être lourdes. Heureusement, les fabricants d'automate proposent de plus en plus d'outils pour simuler le fonctionnement des programmes avec une partie opérative virtuelle comme dans le logiciel Unity Pro de Schneider Electric.

Pourquoi créer une PO virtuelle ?

Lors de la programmation selon la norme IEC 1131-3 « La programmation des API » en FBD (Functional Block Diagram), Ladder (langage à contact), ST (langage structuré texte), SFC (Sequential Functional Chart, proche du Grafcet) ou IL (Instruction List), les étudiants sont amenés à traduire un cahier des charges en langage automate. Plusieurs méthodes sont possibles :

– réaliser ces programmes au tableau (blanc ou noir) sans aucune simulation. Mais il manque alors aux étudiants le côté pratique, et donc le retour d'expérience. Ils se contentent de ce qu'en dit le professeur, ce qui ne leur laisse que peu de souvenirs de leur travail ;

– réaliser ces programmes directement sur des systèmes de l'atelier, ce qui, en théorie, est la meilleure expérience pour les étudiants. Mais les problèmes de programmation sont souvent noyés dans des difficultés de configuration logicielle et matérielle. De plus, la phase de test devient vite complexe car tributaire de la partie opérative (PO) physique avec tous les problèmes et les risques que cela comporte ;

– réaliser ces programmes sur une PO virtuelle qui, elle, réagit au programme, et donc permet aux étudiants d'obtenir un retour d'expérience, sans les problèmes physiques et matériels d'une PO réelle.

Pour créer un simulateur de PO sous Unity Pro, nous allons exploiter le module Écran d'exploitation qui permet de créer des PO de type process chimique avec des cuves, des pompes, des vannes, etc. On peut voir les cuves se remplir ou se vider, mais il est impossible d'y déplacer les objets graphiques représentant un vérin, une caisse, un wagonnet, etc. (ce qui exclut la simulation de machines de production).

L'exemple décrit est fourni à partir des trois fichiers téléchargeables sur le site Éduscol :

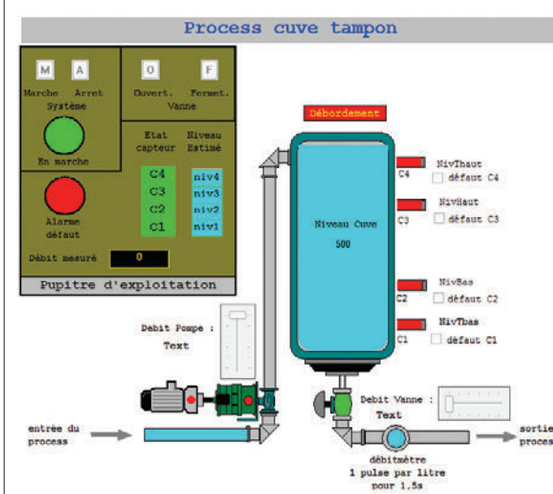
– Unity Pro type archive automate contenant le simulateur programmé : File_1.STA ;

– Exemple corrigé d'une application avec des étudiants de BTS MSP : File_2.PDF ;

– Unity Pro avec la correction de l'exemple d'application : File_3.STA.

Le mot de passe pour déverrouiller les sections en langage ST est « BTSMI » (en majuscules).

Remarque : pour simuler des PO nécessitant des mouvements (déplacement de dessins en translation ou rotation), il faudra utiliser Vijeo Designer simultanément avec Unity. Cette méthode, beaucoup plus souple et performante, est cependant plus complexe d'approche ; elle fera donc l'objet d'un futur article.



1 Allure de notre installation

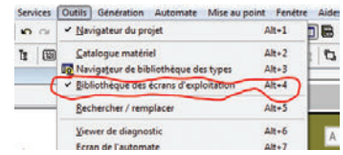
[1] Professeur de maintenance des systèmes, lycée La-Martinière-Diderot, Lyon.

HMI_mess_A	EBOOL			afficheur A HMI
HMI_mess_alarme	EBOOL			voyant alarme HMI
HMI_mess_B	EBOOL			afficheur B HMI
HMI_mess_C	EBOOL			afficheur C HMI
HMI_mess_D	EBOOL			afficheur D HMI
HMI_mess_E	EBOOL			afficheur E HMI
HMI_mess_F	EBOOL			afficheur F HMI
HMI_mess_G	EBOOL			afficheur G HMI
HMI_mess_H	EBOOL			afficheur H HMI
HMI_voyant_marche	EBOOL			voyant marche HMI
I_Cde_Arret_Systeme	EBOOL			Entree API
I_Cde_Fermeture_Vanne	EBOOL			Entree API
I_Cde_Marche_Systeme	EBOOL			Entree API
I_Cde_Ouverture_Vanne	EBOOL			Entree API
I_Debitmetre	EBOOL			Entree API
I_NivBas	EBOOL			Entree API
I_NivHaut	EBOOL			Entree API
I_NivTbas	EBOOL			Entree API
I_NivThaut	EBOOL			Entree API
KM_Mem_En_Service	EBOOL			Memoire API
Q_Marche_Moteur	EBOOL			Sortie API
Q_Ouverture_Vanne	EBOOL			Sortie API

2 Variables E/S API et HMI

Nom	Type	Adresse	Valeur
SIM_C1HS	EBOOL		FALSE
SIM_C2HS	EBOOL		FALSE
SIM_C3HS	EBOOL		FALSE
SIM_C4HS	EBOOL		FALSE
SIM_Debit_Pompe	INT		3
SIM_Debit_Vanne	INT		1
SIM_Debordement	EBOOL		
SIM_Dcleme_s	EBOOL		
SIM_nbr_ds_par_pulse	INT		
SIM_nbr_pulse_en_cours	INT		
SIM_Niveau_Cuve	INT		500
SIM_Niveau_cuve_affichage	INT		50
SIM_pulses_debit	EBOOL		
SIM_Seconde	EBOOL		

3 Variables utiles à la simulation



4 Accès à la bibliothèque des écrans d'exploitation

Démarche de création d'une PO simple

Description de la PO à créer

La PO doit simuler le fonctionnement d'une cuve d'eau servant de tampon entre une pompe et l'utilisateur (plusieurs robinets par exemple) 1. Elle sera donc constituée d'une cuve, d'une motopompe pour remplir la cuve tampon et d'une vanne permettant l'utilisation de l'eau stockée dans la cuve. Afin de « réguler » le niveau d'eau dans la cuve, quatre capteurs devront être créés, ainsi que des boutons-poussoirs (BP) pour la commande.

Avant de réaliser réellement le dessin de la PO (ou de façon simultanée), il faut prévoir les variables automates associées aux boutons-poussoirs et capteurs, aux pompes et vannes et les variables de simulations telles que le niveau de la cuve, les débits de la pompe et de la vanne (réglables dans notre exemple).

Création des variables d'entrées/sorties de l'API

Pour s'y retrouver dans le nom des variables, il est conseillé d'y insérer le rôle de la variable. Ici, quatre types de noms seront utilisés :

– les variables d'entrée API nommées I_xx, par exemple I_Cde_Marche_Systeme, de type EBOOL qui permet de traiter les fronts ;

– les variables de sortie API nommées Q_xx, par exemple Q_Marche_Moteur ;

– les variables d'affichage HMI nommées HMI_xx, par exemple HMI_Voyant_Marche.

Ces trois premières variables seront accessibles aux étudiants, car elles font le lien entre l'API et la PO virtuelle 2.

– les variables utiles uniquement au simulateur nommées SIM_xx, par exemple SIM_Niveau_Cuve, de type INT qui permettra de mémoriser le niveau dans la cuve 3.

Création de la PO

L'étape suivante est de dessiner la PO à l'aide de la bibliothèque d'objet de l'écran d'exploitation et d'associer certains objets aux variables afin d'animer la PO. Ici, l'animation sera de trois types : changement de couleur, affichage de texte ou affichage de bargraphe.

Accès aux outils d'écran d'exploitation

À la création d'un nouvel écran d'exploitation, une barre d'outils associée (en haut de l'écran) permet d'insérer des objets simples (rectangle, cercle…), des champs de saisie, des zones de texte fixe, ou des valeurs de variables. Pour les dessins plus complexes comme la cuve et la pompe, la vanne et les tuyauteries, on fait appel à la bibliothèque des écrans d'exploitation par le menu Outils 4.

On obtient alors la figure 5.

Copie et paramétrage de la cuve depuis la bibliothèque

Dans la bibliothèque, un double-clic sur l'objet cuve ouvre un onglet bibliothèque avec les différentes cuves. Il suffit de copier la cuve voulue et de la coller sur l'écran d'exploitation. Il faut ensuite dissocier les objets composant la cuve (clic droit, puis dissocier, puis sélectionner le rectangle bleu par un clic gauche) afin d'accéder au rectangle bleu (niveau d'eau). Accéder aux propriétés de l'objet par un clic droit, puis Caractéristiques, ce qui ouvre une fenêtre à trois onglets.

Sur la figure 6a, on associe la variable SIM_Niveau_Cuve_Affichage de type INT (variant de 0 à 100) en affichage permanent. Sur la figure 6b, on choisit le type bargraphe, puis, en cliquant sur le bouton >, on peut définir le type de bargraphe (ici, vertical plein vers le haut de 0 à 100).

On peut ensuite regrouper les objets qui composent la cuve.

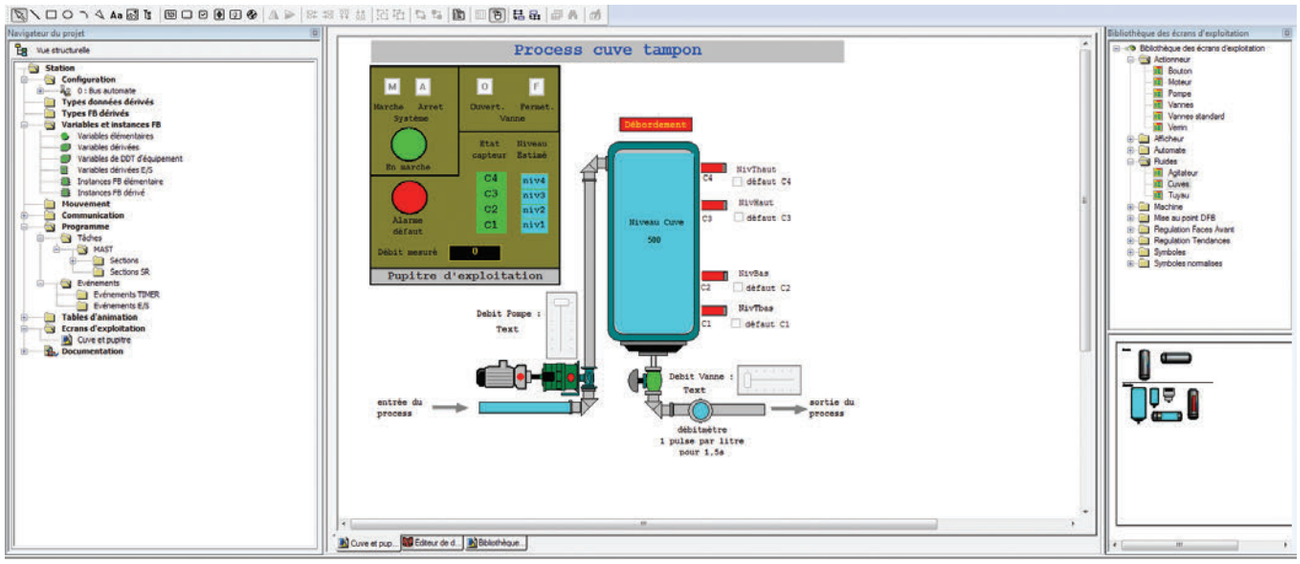
Copie et paramétrage de la pompe depuis la bibliothèque

Comme pour la cuve, on copie une pompe à partir de la bibliothèque, on dissocie la pompe et on paramètre juste l'ellipse rouge qui servira de voyant arrêt 7. Si on déplace l'ellipse rouge, on remarque une ellipse verte en dessous. Il suffit de n'afficher l'ellipse rouge que lorsque la pompe est arrêtée. L'ellipse verte sera visible si la pompe marche.

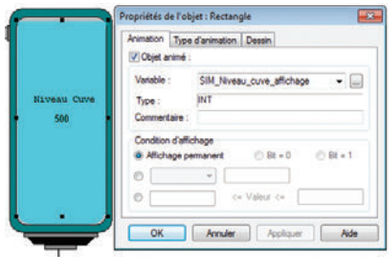
On associe l'affichage de l'ellipse rouge à Q_Marche_Moteur lorsque la variable EBOOL est à 0 (bit = 0).

Création d'un bouton-poussoir à partir de la barre d'outils

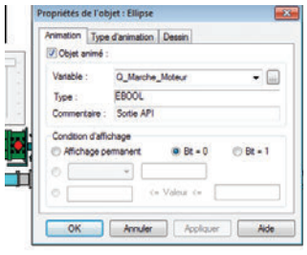
Pour le bouton-poussoir de mise en marche du système, on utilise l'objet Bouton de commande de la barre d'outils. On associe l'animation à la variable I_Cde_Marche_Systeme 8a et le pilotage à la même variable



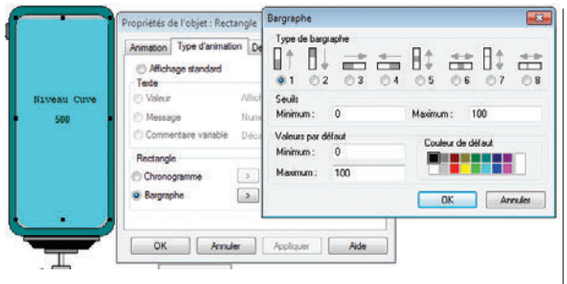
5 Écran en mode création de PO



6a Caractéristiques rect. bleu – onglet Animation



7 Caractéristiques ellipse rouge – onglet Type d'animation



6b Caractéristiques rect. bleu – onglet Type d'animation

afin qu'elle soit pilotée par un appui sur le BP **8b**. On peut aussi créer un interrupteur en cochant l'option avec accrochage de l'onglet pilotage.

Réalisation des autres parties du simulateur

Pour créer les curseurs de réglage de débit, utiliser un objet Curseur de la barre d'outils. Pour afficher la valeur du niveau, utiliser un objet Text. Pour les capteurs, on anime de simples rectangles avec des couleurs. Pour les défauts de capteurs, on peut utiliser l'outil Cases à cocher, comme dans l'exemple. Le fichier Exemple Unity Pro et plus particulièrement les caractéristiques des différents objets de l'écran d'exploitation donnent un aperçu de ce que l'on peut en faire.

Protection du programme du simulateur

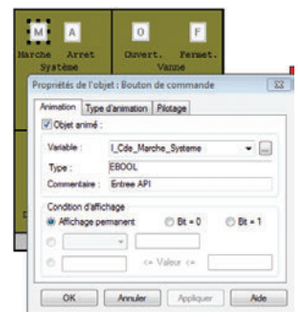
Afin d'éviter que les étudiants ne détruisent le programme du simulateur par inadvertance en réalisant leur propre programme, verrouiller les sections de programme du simulateur. Pour cela, il suffit de définir les sections en mode de protection Pas de lecture ni d'écriture, comme indiqué sur la figure **9**. Puis, par un clic droit sur Station dans le navigateur de projet, puis dans Propriétés, activer la protection en entrant un mot de passe **10** (dans notre exemple : « BTSMI »).

Programmation du simulateur

On peut utiliser le langage de son choix pour le code du simulateur (le ST est tout indiqué car plus souple et plus compact pour ce type de programmation).

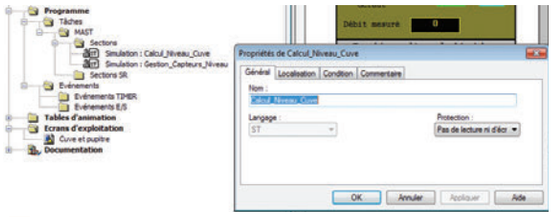
Deux remarques importantes avant de se lancer dans le code du simulateur :

– la logique du simulateur est très différente de celle du programme API dans le sens où, ici, les entrées seront les sorties API (commande moteur et vanne), et les sorties les entrées API (capteurs). Il faut donc raisonner en comportemental au niveau de la PO. Si la pompe est active, le niveau augmente ; si le niveau augmente, les capteurs changent d'états ;

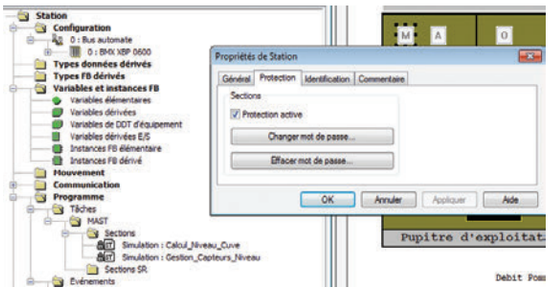


8a Caractéristiques du BP marche – onglet « Animation »

8b Caractéristiques du BP marche – onglet « Pilotage »



9 Paramétrer une section en mode protégé (cadenas)



10 Activation/désactivation de la protection

– pour éviter que les objets s'animent à une vitesse qui dépend du processeur, il faut définir une base de temps qui cadencera l'évolution de la PO. Dans notre exemple, le niveau augmente tous les dixièmes de seconde si la pompe fonctionne. De même, le niveau baissera tous les dixièmes de seconde si la vanne est ouverte. L'animation du niveau prend en compte les débits.

Ce qui donne les programmes des deux encadrés a et b ; les listings étant commentés, ils sont compréhensibles.

Exemple d'utilisation avec les étudiants

Nous allons voir comment utiliser le simulateur sur un cahier des charges simple aboutissant sur quelques équations logiques à programmer en FBD par exemple dans notre cas.

Cahier des charges destiné aux étudiants (lié à la PO créée)

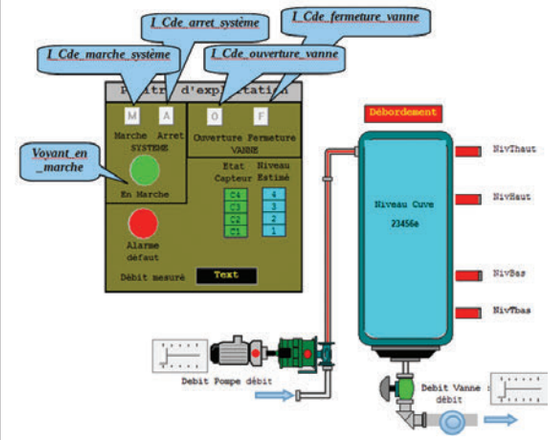
On souhaite automatiser le maintien d'un niveau dans une cuve 11. Le remplissage se fait par une motopompe de débit de 2 l/s et la vidange par une vanne avec un débit de 3 l/s.

Cahier des charges

Le bouton-poussoir M (I_Cde_Marche_Systeme) met en service l'installation et le BP A (I_Cde_Arret_Système) arrête l'installation. La mémorisation de la mise en service se fait par la variable KM_Mem_en_service.

La pompe se met en marche si elle est en service et que le niveau descend en dessous du capteur Niveau Bas (I_NivBas). La pompe s'arrête si le niveau atteint le niveau très haut (I_NivThaut).

La vanne est commandée manuellement par les BP O (I_Cde_Ouverture) et F (I_Cde_Fermeture), mais ne peut pas s'ouvrir si le niveau est très bas (I_NivTbas).



11 Schéma synoptique du système

```
SIM_Seconde := %S6;
SIM_Dixieme_s := %S5;
 (* initialisation des variables au premier cycle *)
if (%S13) then
        SIM_Debit_Pompe:=1;
        SIM_Debit_Vanne:=2;
        SIM_nbr_ds_par_pulse:=0;
        SIM_nbr_pulse_en_cours:=0;
end_if;


If RE (SIM_Dixieme_s) then
        (* remplissage ou vidange de la cuve*)
        If Q_Marche_Moteur and SIM_Niveau_Cuve < 1000 then
                SIM_Niveau_Cuve := SIM_Niveau_Cuve + SIM_Debit_Pompe;
        end_if;

        If Q_Ouverture_Vanne and SIM_Niveau_Cuve > 0 then
                SIM_Niveau_Cuve := SIM_Niveau_Cuve - SIM_Debit_Vanne;
                SIM_nbr_ds_par_pulse := 15/SIM_Debit_Vanne;
        ELSE SIM_nbr_ds_par_pulse := 0;
        end_if;
        (* generation des pulses du débimètre en fonction du débit vanne *)
        if SIM_nbr_ds_par_pulse>0 then
                If (SIM_nbr_pulse_en_cours=SIM_nbr_ds_par_pulse) then
                        SIM_pulses_debit:=true;
                        SIM_nbr_pulse_en_cours:=0;
                else
                        SIM_pulses_debit:=false;
                        SIM_nbr_pulse_en_cours:=SIM_nbr_pulse_en_cours+1;
                end_if;
        else
                SIM_pulses_debit:=false;
        end_if;
End_If;
 (*calcul du débordement*)
if (SIM_Niveau_Cuve>=1000) then
        SIM_debordement:=true;
        SIM_Niveau_Cuve:=1000;
else
        SIM_debordement:=false;
end_if;
 (* affichage niveau cuve bleu*)
SIM_Niveau_cuve_affichage:=SIM_Niveau_Cuve/10;
I_Debitmetre:=SIM_pulses_debit;
```

a Simulation du comportement de la cuve (pompe et vanne)

```
If ((SIM_Niveau_Cuve > 100)and(not SIM_C1HS)) then
        Set (I_NivTbas);
else
        Reset (I_NivTbas);

End_if;

If ((SIM_Niveau_Cuve > 300)and(not SIM_C2HS)) then
        Set (I_NivBas);
else
        Reset (I_NivBas);
End_if;

If ((SIM_Niveau_Cuve > 700)and(not SIM_C3HS)) then
        Set (I_NivHaut);
else
        Reset (I_NivHaut);
End_if;

If ((SIM_Niveau_Cuve > 900)and(not SIM_C4HS)) then
        Set (I_NivThaut);
else
        Reset (I_NivThaut);
End_if;

(*
NivTBas := niveau_cuve > 100;
NivBas := niveau_cuve > 300;
NivHaut := niveau_cuve > 700;
NivTHaut := niveau_cuve > 900;
*)
```

b Simulation du comportement de capteurs

Équations trouvées à partir du cahier des charges

KM_Mem_en_service=/I_cde_arret_systeme.(I_cde_marche_systeme+KM_Mem_en_service)
HMI_Voyant marche=KM_Mem_en_service
Set(Q_Marche_Moteur) =/I_NivBas.KM_mem_en_service
Reset(Q_Marche_Moteur) =I_NivTHaut+/KM_mem_en_service
Q_Ouverture_Vanne=I_NivTbas./I_cde_fermeture_vanne.(I_cde_ouverture_vanne+Q_ouverture_vanne)

Programme créé par les étudiants en FBD

Le listing, ici en FBD, reprend les équations logiques précédentes sous forme de logigramme. La fonction de recopie pour l'équation du voyant de marche est réalisée par la fonction MOVE. Une section de programme permet de gérer la mise en service 12a et une autre section gère la pompe et la vanne 12b.

Comment lancer la simulation ?

Pour lancer la simulation, cinq étapes doivent être respectées :

– régénérer le projet (compilation) par le menu Génération, puis Régénérer tout le projet ;

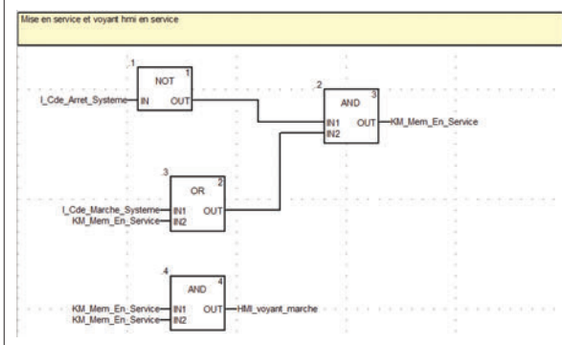
– se connecter à l'automate en mode simulation, menu Automate, puis Mode simulation. Ensuite menu Automate, puis Connexion ;

– transférer le programme dans l'automate virtuel, menu Automate, puis Transférer le projet vers l'automate ;

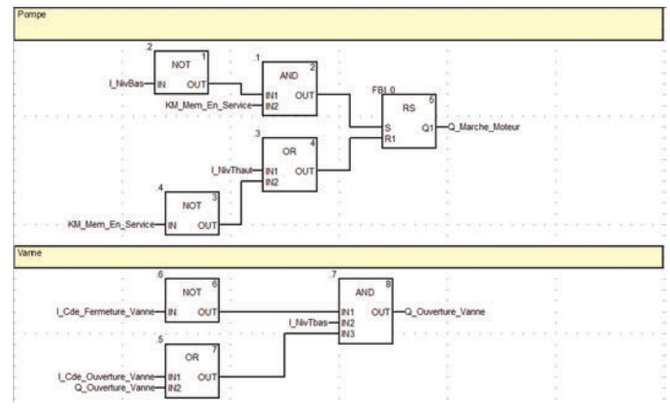
– passer en mode run ;

– sélectionner l'écran d'exploitation et valider l'écriture des variables par F7 ou en cliquant sur l'icône la plus à droite de la barre d'outils de l'écran d'exploitation.

À ce stade, le système virtuel peut être mis en service en appuyant sur le BP marche et en ouvrant la vanne. On observe un fonctionnement calqué sur un système réel. ■



12a FBD de la mise en service



12b FBD de la pompe et de la vanne

Retour d'expérience

Ce type de simulateur très simple permet d'aborder les différents langages avec une résolution de courte durée, et peut être intégré dans une séance de TP d'environ 2 heures Les étudiants peuvent ainsi vérifier en autonomie la fonctionnalité de leur programme, corriger leurs éventuelles erreurs de raisonnement ou de transcription dans le langage API choisi.

En revanche, il ne permet pas de traiter les modes de marche et d'arrêt avec des traitements hiérarchisés à l'aide, par exemple, de SFC synchronisés. Un prochain article proposera un simulateur de traitement de surface (sous Vijeo Designer) pouvant gérer plusieurs modes de marche (manuel, automatique, initialisation, arrêt d'urgence…) et qui permet de traiter des cahiers des charges débouchant sur des SFC (ou séquenceurs) à plusieurs niveaux avec synchronisation, figeage, réinitialisation des séquenceurs et toute autre structure évoluée.