Développement d'un connecteur logiciel pour l'apprentissage de l'automatisme

Philippot Alexandre, Stéphane Lecasse, Bernard Riera, François Gellot

Développement d'un connecteur logiciel pour l'apprentissage de l'automatisme

Alexandre Philippot[1], Stéphane Lecasse[1], Bernard Riera[1], François Gellot[1]

[1] CReSTIC (EA3804), Université de Reims Champagne-Ardenne,
Moulin de la Housse, 51687 Reims, France

RESUME : L'apprentissage de l'automatisme s'adresse aujourd'hui à un public d'étudiants ayant pour la plupart avant tout un *background* informatique. Ils/elles passent par des phases d'apprentissage de langages compilés et/ou interprétés. Il est compliqué alors pour eux de passer d'une réflexion informatique avec un langage venant du monde de l'IT (Information Technology) vers la programmation d'Automates Programmable Industriel (API), au comportement cyclique, synchrone et aux langages normalisés (IEC 61131-3) issu du monde de l'OT (Operational Technology). Ce papier présente une proposition de mise en place d'un connecteur logiciel entre ces deux mondes aux travers de l'utilisation d'un logiciel de simulation de Parties Opératives Factory I/O (realgames.co) et du langage Python.

Mots clés : Automatisme, Simulation, Information Technology, Operational Technology.

1 INTRODUCTION

En 2013, lors du CETSIS de Caen, le Centre de Recherche en STIC (CReSTIC EA3804) a présenté un interpréteur de Grafcet pour l'enseignement et la recherche [1]. L'outil, l'Interpréteur G7, permet de simuler des Grafcets sans se soucier d'une transposition dans un langage de programmation des Automates Programmables Industriels (API) de la norme IEC 61131-3. Cet outil s'est révélé très pertinent dans le cadre des cours où les étudiants se familiarisaient avec la modélisation à l'aide de grafcet. Ils pouvaient ainsi visualiser, en couplant l'outil avec le logiciel de simulation de partie opérative ITS PLC développé par la société REAL GAMES (realgames.co), le résultat et l'exécution de leur modélisation.
Cependant, avant d'arriver à établir une spécification Grafcet et comprendre le fonctionnement d'un API, l'enseignement des systèmes automatisés doit passer par plusieurs phases d'apprentissage. Par exemple, dans la Licence Electronique, Energie Electrique, Automatique (EEEA) de l'Université de Reims Champagne-Ardenne, ces différentes étapes s'expriment sur les 3 années à travers :

- 2 modules de « Logique » en année 1 pour un total de 42h ;
- 1 module « Algorithmique et langage C » de 30h en semestre 3 et 1 module « TP longue durée informatique » de 30h également en semestre 4 ;
- 3 modules de « Systèmes logiques et Automatismes » d'un total de 63h.

L'objectif du premier module de logique est la conception et réalisation des systèmes logiques combinatoires de base et suit le programme suivant :

- Numération, codage.
- Fonctions logiques élémentaires : définition, simplification, Karnaugh.
- Synthèse d'un système combinatoire, implantation.
- Logique séquentielle : mémoires, bascules RS, D, JK, registres, compteurs.
- Technologie de fabrication des circuits logiques combinatoire de base et leurs caractéristiques.

Le second module logique vise quant à lui à concevoir et réaliser des systèmes séquentiels synchrones et asynchrones, ainsi qu'à générer des fonctions logiques en utilisant des mémoires et des réseaux logiques programmables. Le programme se compose de l'analyse et la synthèse d'un système séquentiel, de la technologie des circuits ainsi que de l'initiation à la logique programmable.

Le module « Algorithmique et langage C » est une initiation aux structures matérielles et au langage informatique. Il permet aux étudiants de voir la production de programmes, les compilateurs et interpréteurs. Il introduit notamment les structures de données et algorithmes associés (tables, listes, piles, files). Le langage C est utilisé comme élément de base pour la programmation structurée. Le module « TP longue durée informatique » sert quant à lui de réalisation d'un programme informatique d'après un cahier des charges.

En licence 3ème année, le premier module « Systèmes logiques et Automatismes » effectue des rappels de logique afin d'étudier, analyser et synthétiser les systèmes séquentiels synchrones et asynchrones. Le deuxième ne fait que du cours magistral et des travaux dirigés en automatismes séquentiels. On y retrouve :

- Introduction à l'automatisme
- Formalisation d'un système à l'aide d'un grafcet : définition, concept de base.
- Lien bascules, automate, grafcet : implantation câblée du grafcet

- Implantation programmée du grafcet : mise en œuvre d'un automate programmable industriel.

Le troisième module se concentre uniquement sur des travaux pratiques autour de la logique programmée et de la programmation d'un automate programmable industriel.

De ce programme annoncé (fig. 1), il est rapide de constater qu'il est compliqué pour les étudiants de passer d'une réflexion informatique avec un langage venant du monde de l'IT (Information Technology) vers la programmation d'Automates Programmables Industriels (API), au comportement cyclique, en langage normalisé (IEC 61131-3) issu du monde de l'OT (Operational Technology) [2, 3].

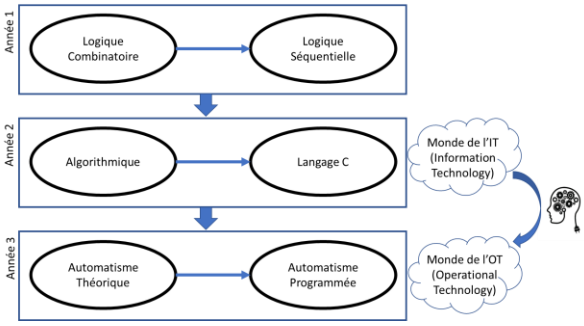

fig 1 : Mondes de l'IT et de l'OT

Par ailleurs, le module « TP longue durée informatique » ne s'applique qu'à du logiciel alors que les TP d'automatismes vont s'appliquer sur des parties opératives réelles.
Afin de combler ce gap, nous proposons d'exploiter le SDK Engine I/O fourni avec Factory I/O pour permettre aux étudiants de programmer des contrôleurs logiques en langage Python sous l'environnement de développement intégré IDLE. Par ailleurs, un serveur websocket a été mis en place pour disposer d'une page HTML afin de piloter le système virtuel en mode manuel (fig. 2).

fig 2 : Connecteur logiciel

La deuxième partie du papier détaille la mise en place du connecteur logiciel et de l'installation du Websocket. La troisième partie est un développement en cours qui illustre un des benchmarks utilisé sous Factory I/O sur une maquette réelle.

2 CONNECTEUR LOGICIEL POUR L'APPRENTISSAGE DE L'AUTOMATISME

2.1 Factory I/O

Factory I/O (https://factoryio.com/) est un logiciel de simulation 3D de systèmes industriels pour la formation aux automates programmables industriels (API) et aux technologies de l'automatisation. Sa première version a été notamment introduite durant le CETSIS 2017 [4]. Pour rappel, il est destiné à l'enseignement technique et professionnel (lycées, IUT, BTS, écoles d'ingénieurs, licences, master…) et est le successeur du logiciel ITS PLC [5]. Factory I/O permet de construire ses propres installations à partir d'une bibliothèque de composants standards que l'on trouve dans le monde industriel et propose également 20 systèmes prêts à l'emploi (cf. figure 3) permettant de couvrir un large éventail d'applications de commande.

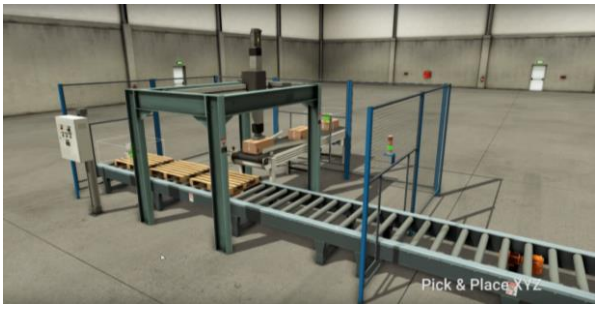

fig 3 : Système « Pick and Place » de Factory I/O

Factory I/O intègre des drivers de communication permettant de s'interfacer directement avec des Automates Programmables Industriels (API de marque SIEMENS, Allen-Bradley, Schneider Electric), avec des simulateurs d'automates au travers d'une connexion Modbus TCP ou d'une connexion à un serveur OPC (DA ou UA) ou tout autre type de matériel (API, microcontrôleur…) au travers des boitiers d'E/S logiques ou analogiques (DAQ Advantech 4750 et 4704).
Enfin, il est également possible d'interfacer ses propres drivers de communication au moyen du SDK Engine I/O fourni permettant par exemple une connexion simple avec MATLAB ou LabVIEW.

2.2 Engine I/O

Engine I/O est disponible depuis un lien de téléchargement GitHub fourni dans la page de documentation pour le développement d'outils SDK de Factory I/O (https://docs.factoryio.com/sdk/). Une fois débloqué, il permet de visualiser les différentes variables de la scène actuellement en *Run* sur Factory I/O grâce à une dll installé dans le même dossier que le fichier python. Ce tableau indique notamment le type et l'adresse de la variable échangée (fig. 4).

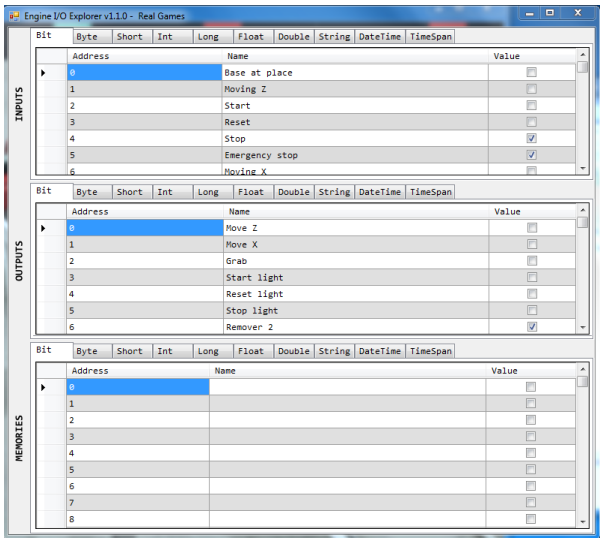

fig 4 : Vue des variables d'une scène Factory I/O depuis le connecteur logiciel Engine I/O

C'est depuis ce connecteur logiciel, que Python va pourvoir interagir sur le système automatisé simulé de Factory I/O.

2.3 Python

Python est un langage de programmation interprété multiplateforme et placé sous une licence libre (https://www.python.org/). Il fonctionne sur la plupart des plateformes informatiques et utilise une syntaxe simple qui en fait un outil d'initiation aux concepts de base de la programmation apprécié par de nombreux enseignants car fortement inspiré d'autres langages comme le langage C.

Pour notre utilisation, nous utiliserons la version 3.7.6 (32 bits). Avant de débuter la programmation d'une scène, il convient d'installer les différents environnements et bibliothèques de communication. Pour la communication avec Engine I/O, la commande « pip install pythonnet » doit être exécutée depuis l'invite cmd.exe mais aussi sa mise à jour par « pip install –upgrade pip ». Pythonnet (Python pour .NET) permet une intégration presque parfaite avec le CLR (Common Language Runtime) de .NET 4.0+ sous Windows. Python pour .NET fournit un puissant outil de script d'application pour les développeurs .NET. Cet outil permet de créer des applications .NET ou des applications complètes en Python, en utilisant des composants .NET écrits dans n'importe quel langage qui cible le CLR (C#, VB.NET, F#, C++/CLI). Python pour .NET est actuellement compatible et testé avec les versions 2.7 et 3.5-3.7 de Python.

Par ailleurs, pour l'installation et la communication du serveur web en mode manuel, l'invite de commande « pip install SimpleWebSocketServer » est lancée.

Il est possible ensuite d'éditer avec l'environnement de développement intégré pour le langage Python IDLE 32 bits le fichier de la scène à exécuter (fig. 5). La structure de ce fichier est composée de 4 zones (sans compter l'import des bibliothèques classiques) :

1. Import de l'adressage des variables à travers la dll EngineI/O.
2. Déclaration des entrées/sorties avec mappage.
3. Déclaration des variables locales avec initialisation des valeurs. Cette section se termine par une première mise à jour des variables
4. Reproduction du programme cyclique de l'API dans une boucle While avec mise à jour de la valeur des variables à chaque cycle. Ce programme traduit la spécification Grafcet obtenue par les étudiants par la méthode dite de l'auto-maintien [6].

Le lancement de ce script depuis IDLE est réalisé par F5 (Run/Run Module) qui ouvrira Python 3.7.6 Shell jusqu'à son interruption par Ctrl+C (Shell/Interrupt Execution).

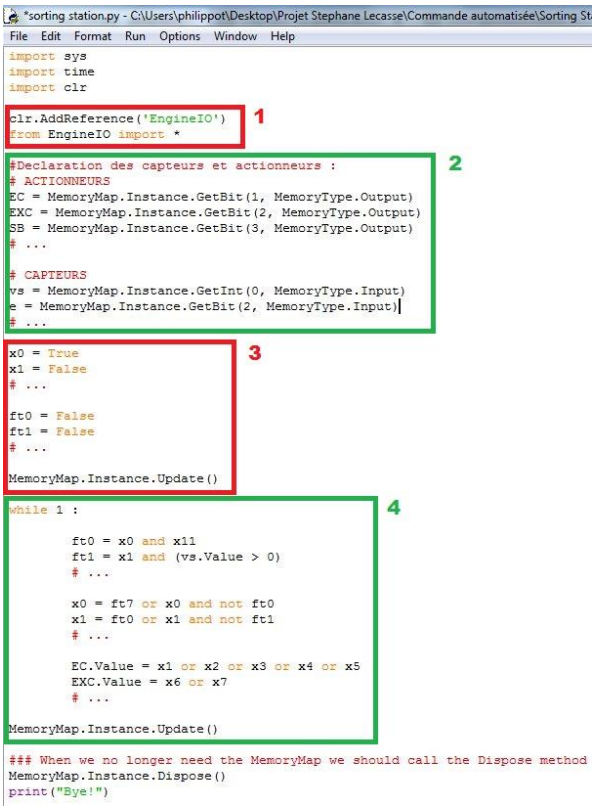

fig 5 : Structure du fichier Python

2.4 Serveur Websocket de page HTML pour un mode manuel

En termes d'apprentissage, il est essentiel pour l'étudiant de comprendre le comportement d'un système sous la forme d'un mode manuel où il est acteur des ordres envoyés. La création d'un mode manuel par serveur Websocket demande de modifier la structure du fichier python (fig. 6). Celui-ci est composé de 5 parties (sans compter l'import de toutes les bibliothèques nécessaires) :

1. La déclaration sous forme de tableau des entrées/sorties du système avec mappage, mais aussi la déclaration des anciennes variables à l'instant précédent. Ainsi que la définition de

boutons de type « switch » pour la commande manuelle des actionneurs

2. Création du serveur Web
3. Ecriture des valeurs des boutons dans la mémoire des actionneurs
4. Mise en tableau des valeurs des capteurs avec lecture en temps réel par l'intermédiaire d'une instruction parallèle *thread*.
5. Exécution du serveur en localhost.

```
import sys
import time
import clr
import threading
clr.AddReference('EngineIO')
from EngineIO import *

#Declaration des capteurs et actionneurs :                                  1
actionneurs = {}
actionneurs['Load'] = MemoryMap.Instance.GetBit(0, MemoryType.Output) #...
capteurs = {}
capteurs['High.sensor'] = MemoryMap.Instance.GetBit(5, MemoryType.Input) #...
oldvalues = {}
oldvalues['High.sensor'] = MemoryMap.Instance.GetBit(5, MemoryType.Input).Value #...

def action_switch(outputs):
        actionneurs[outputs].Value = not actionneurs[outputs].Value
    MemoryMap.Instance.Update() #MàJ de la mémoire de EngineI/O

#Création serveur Web                                                       2
from SimpleWebSocketServer import SimpleWebSocketServer, WebSocket
class SimpleEcho(WebSocket):

#Ecriture des valeurs dans la mémoire pour les actionneurs                  3
    def handleMessage(self):
        action_switch(self.data)
        MemoryMap.Instance.Update()

#Mise en tableau des valeurs M-1 des capteurs                               4
    def lecture_capteurs(self):
        while 1:
            MemoryMap.Instance.Update()
            for capteur in capteurs :
                if capteurs[capteur].Value != oldvalues[capteur] :
                    oldvalues[capteur]=capteurs[capteur].Value
                    self.sendMessage(capteur)
    def handleConnected(self): #lecture des capteurs en tps réel
        for capteur in capteurs :
            if (capteurs[capteur].Value):
                self.sendMessage(capteur)
        try:
            thread1 = threading.Thread(target = self.lecture_capteurs, args = ()) #programme parrallèle
            thread1.start()
        except Exception as err:
            print(err)

server = SimpleWebSocketServer('', 443, SimpleEcho) # Port 443 en localhost   5
server.serveforever()
```

fig 6 : Structure du fichier Python pour le mode manuel

L'instruction javascript sur le fichier html est défini en figure 7. L'interface de développement obtenue pour le système de tri de Factory I/O est illustrée en figure 8. On y voit les voyants des différents capteurs ainsi que les boutons de mise en marche des actionneurs.

```
<script language="javascript" type="text/javascript">

   websocket = new WebSocket("ws://localhost:443/");

   websocket.onmessage = function(evt) { etats(evt) };

/* Envoi de la commande des actionneurs au serveur */
    function sendText(message)
    {
        websocket.send(message);
    }

/*Lecture de l'état des capteurs*/
    function etats (evt)
    {
      console.log(evt);
      document.getElementById(evt.data).checked = !document.getElementById(evt.data).checked;
    }

</script>
```

fig 7 : Instruction serveur Websocket en HTML

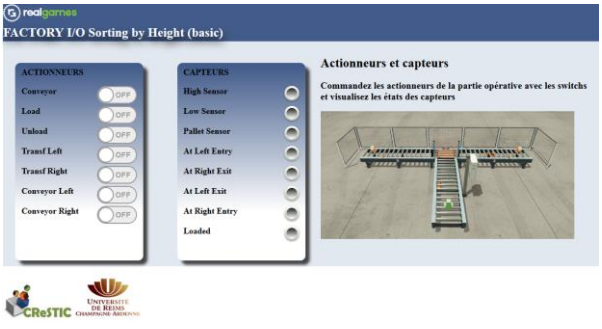

fig 8 : Interface web pour le système de tri

3 MISE EN PLACE D'UNE MAQUETTE REELLE

En termes d'application pédagogique, il convient de mélanger les mondes virtuels et réels [7, 8]. Dans ce cadre, une maquette du système de tri de caisses de Factory I/O a été réalisé sur la base de briques LEGO (fig. 9).

L'interaction est nécessaire par l'intermédiaire de cartes d'acquisition de données. L'utilisation d'une carte de type Raspberry Pi utilisé comme un API et exploitant ses E/S directement (https://www.framboise314.fr/unipi-un-automate-industriel-avec-le-raspberry-pi/) est en cours de développement pour être présenté durant le colloque. Ainsi, la maquette pilotée par un contrôleur réel permettra une mise en service virtuelle lors de l'utilisation de Factory I/O comme « jumeau numérique ».

.

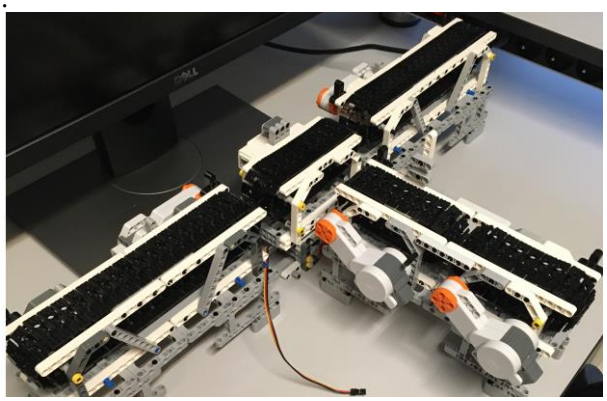

fig 9 : Maquette LEGO du système de tri

4 RETOUR D'EXPERIENCE AVEC HOME I/O ET FACTORY I/O

FACTORY I/O est utilisé à l'URCA en TP de logique séquentielle dans plusieurs formations parmi lesquelles la licence 3 SPI EEA (56 étudiants), la 1ère année de cycle ingénieur A2I (URCA-UTT, 48 étudiants). Le retour d'expérience réalisé pour le moment porte sur l'utilisation de FACTORY I/O et non encore sur le connecteur python. La durée des séances de TP était similaire pour les 2 formations (3 heures), les objectifs pédagogiques identiques (initiation à la logique combinatoire et séquentielle), les enseignants différents pour les 2 filières, et les sujets de TP différents. L'évaluation a été faite au travers de 3 critères : l'utilité pour l'adéquation entre l'objectif d'apprentissage et l'atteinte de cet objectif, l'utilisabilité pour la facilité d'utilisation (la maniabilité) et l'acceptabilité pour s'approprier l'outil. L'analyse statistique des résultats donne :

- Sur la facilité d'utilisation de FACTORY I/O : 70% d'avis de relativement facile à très facile.
- Sur l'aspect ludique : 56% des répondants sont d'accord pour dire que ce logiciel est ludique.

86% des répondants considèrent l'aspect ludique comme étant motivant.

- Sur l'utilisabilité : (i) une qualité bonne ou excellente pour 99% des répondants ; (ii) une facilité à comprendre le fonctionnement de 90% de relativement facile à très facile.

- Pour l'utilité : toutes les dimensions (partie opérative, partie commande, logique séquentielle, capteurs, actionneurs) sont travaillées. Entre 93% et 100% des répondants affirment avoir travaillé au moins partiellement ces concepts.

5 CONCLUSION

L'application de ce développement sur un échantillon d'étudiants n'a pas encore été mise en œuvre sur l'utilisation du connecteur de logiciel. Une expérimentation avec suivi et questionnaire afin d'analyser le retour d'expérience des étudiants sera prochainement mise en place. Elle est en effet nécessaire sur le côté utilisation de Python et compréhension de la connexion des briques technologiques.
Concernant la maquette LEGO, la situation sanitaire depuis un an n'a pas permis son utilisation.

Bibliographie

[1] D. Annebicque, A. Philippot, F. Gellot et B. Riera. Un interpréteur de GRAFCET pour l'enseignement et la recherche. 10ème Colloque Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 2013), Caen, mars 2013.

[2] Arango, F., Aziz, E-S., Esche, S.K., Chassapis, C., «A review of applications of computer games in education and training », *Frontiers in Edu. Conf.. Pp. T4A.1-4A.6, 2008.*

[3] Bainbridge, L., « Ironies of automation ». *Automatica, vol 19, N°6, pp.775-779.*

[4] Bernard Riera, Romain Pichard, Alexandre Philippot, Ramla Saddem, François Gellot, David Annebicque, Fabien Emprin. HOME I/O et FACTORY I/O : 2 logiciels innovants de simulation de PO pour la formation à l'automatique. 12e Colloque consacré à l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes 2017.

[5] Riera, B., Vigario, B., Chemla, J-P., Correia, L., Gellot, F., « 10 ans de Maquettes Virtuelles pour l'enseignement des automatismes : de WINSIM en 1998 à ITS PLC PE en 2008 », *J3eA 8 (HORS SÉRIE 1) 1004 (2009).*

[6] William S. Levine. « The Control Handbook, Second Edition ». CRC Press, December 2010. ISBN 9781420073669.

[7] Corlu, M. S., Capraro, R. M., & Capraro, M. M. (2014). Introducing STEM education: Implications for educating our teachers in the age of innovation. Education and Science, 39(171), 74-85.

[8] Gonzalez H.B., Kuenzi J.J. (2012). Science, technology, engineering and mathematics (STEM) education: A primer Congressional Research Service, Washington: DC (2012).