J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes
J3eA 24 (2025) 9
https://doi.org/10.1051/j3ea/20250009
https://www.j3ea.org

Automatisation d'un robot pédagogique sous IEC 61499

Alexandre Philippot[1]*, Stéphane Lecasse[1], Alexandre Parant[2], Laurent Arcese[1], François Gellot[1]

[1] Université de Reims Champagne-Ardenne, CReSTIC, Reims, France
[2] CESI, CESI LINEACT, Reims, France

*Auteur de correspondance : alexandre.philippot@univ-reims.fr

Résumé : *Cet article explore l'automatisation d'un robot pédagogique Niryo Ned2 en utilisant la norme IEC 61499 et le logiciel EcoStruxure Automation Expert (EAE) de Schneider Electric. L'idée est de développer une plateforme de formation pour les étudiants et les techniciens, adaptée aux exigences de programmation d'automatisme et de robotique de l'Industrie du futur. La plateforme comprend un robot collaboratif 6 axes et un convoyeur, connectés via Modbus TCP. L'environnement EAE permet la programmation événementielle en IEC 61499, facilitant la gestion des tâches complexes et la communication entre les différents composants. Une interface homme-machine (IHM) basée sur Node-RED est également mise en place pour une meilleure interaction utilisateur.*

Mots clés : robotique, IEC 61499, automatisme, protocole Modbus.

Abstract: *This article explores the automation of a Niryo Ned 2 educational robot using the IEC 61499 standard and Schneider Electric's EcoStruxure Automation Expert (EAE) software. The idea is to develop a training platform for students and technicians, adapted to the programming requirements of automation and robotics in the Industry of the Future. The platform includes a 6-axis collaborative robot and a conveyor, connected via Modbus TCP. The EAE environment enables event-driven programming in IEC 61499, facilitating the management of complex tasks and communication between different components. A human-machine interface (HMI) based on Node-RED has also been implemented for enhanced user interaction.*

Keywords: *robotics, IEC 61499, automation, Modbus protocol.*

1. Introduction

Dans la seconde moitié du XXe siècle, la production de masse a commencé à se généraliser dans le secteur manufacturier. En effet, la transition urbaine étant engagée dans de nombreux pays, la demande des ménages en nouveaux produits afin d'améliorer leur confort de vie ne faisait que croître. Si les concepts de la production de masse étaient déjà existants depuis quelques décennies, l'utilisation des robots intégrés dans les chaînes de production a sensiblement changé la donne dans la seconde moitié du XX^{e} siècle [1]. Ces robots permettaient d'accroître la productivité et la rentabilité de l'entreprise en leur dédiant les tâches les plus répétitives et dangereuses pour l'Humain. Si les premiers robots réalisaient des tâches relativement basiques et ne nécessitaient pas de formation poussée de la part des opérateurs pour les utiliser, ce n'est plus le cas concernant les robots modernes. Ces systèmes cyber-physiques de production sont en effet de plus en plus complexes, et souvent ces robots travaillent en coopération et de manière synchronisée afin de réduire au maximum le temps de production total [2]. De plus, l'évolution des concepts de l'Industrie 4.0 vers l'Industrie 5.0, en redéfinissant notamment la place de l'Humain dans ces systèmes manufacturiers, est en cours, et requiert de nouveau l'acquisition de nouvelles compétences [3].

Dans ce contexte, le besoin des industriels à recruter des techniciens et des ingénieurs qualifiés ne fait que s'accroître. Les compétences des sciences de l'ingénieur à acquérir pour concevoir et développer des systèmes robotisés, sont nombreuses et pluridisciplinaires. En effet, il est souvent nécessaire d'avoir une solide formation en robotique, en automatisme/automatique ainsi qu'en informatique industrielle. Néanmoins, l'opérateur terrain de tels systèmes doit être en mesure de contrôler ces systèmes robotisés à l'aide d'outils de programmation sans nécessairement posséder un haut niveau de qualification en robotique. Dans ce cadre, des plateformes robotiques peuvent être mises en œuvre afin de former les étudiants à l'utilisation de tels robots.

Par ailleurs, pour répondre aux besoins de flexibilité de production et être en capacités de produire une grande variété de produits sur plusieurs générations, les systèmes de commande doivent être facilement modifiables. Cependant, les systèmes de production sont des sont majoritairement programmés à partir des langages de la norme IEC 61131-3 qui possèdent une structure monolithique difficile à faire évoluer [4]. De plus, les incompatibilités nombreuses entre les différents automates programmables limitent les systèmes autant en flexibilité qu'en reconfigurabilité. C'est notamment pour pallier ce type de problème que la série de normes IEC 61499 a été créée [5]. Elle doit permettre le développement de système de commande distribuée par l'implémentation d'applications sur un ensemble de contrôleurs partageant le même réseau.

C'est dans ce contexte que nous proposons d'introduire la programmation d'un robot Niryo Ned 2 par l'intermédiaire du logiciel EcoStruxure Automation Expert (EAE) de Schneider Electric[1] en IEC 61499. Cette proposition sera introduite dans un module d'automatisme de la formation ingénieure A2I (Automatique et Informatique Industriel) de l'EiSINe (École d'ingénieurs en Sciences Industrielles et Numérique) de l'Université de Reims Champagne-Ardenne (URCA). Le public étudiant visé est de niveau BAC+5, et donc ayant déjà un niveau de connaissance en automatisme dite « classique » (IEC 61131-2) de bon niveau. À l'heure de la rédaction de ce papier, aucune expérimentation n'a pu être menée. Le créneau envisagé étant

[1] https://www.se.com/fr/fr/product-range/23643079-ecostruxure-automation-expert#overview

en fin de second semestre. La seconde section présente succinctement l'architecture de la plateforme d'expérimentation. La communication et le programme en IEC 61499 sont décrits en section 3 avant de discuter des difficultés rencontrées en section 4. Le papier conclut notamment avec des perspectives de développement élargissement le cadre d'étude.

2. Architecture Robotique en IEC 61499

La plateforme utilisée dans ce travail et présentée dans ce papier est constituée d'un robot Niryo Ned 2[2] 6 axes de type collaboratif et d'un convoyeur (Figure 1). Ce robot académique est destiné à la découverte de la robotique pour des apprenants de collèges, lycées, universités et écoles d'ingénieurs mais évidemment sur des thématiques diversement complexes. Ainsi, il est possible d'utiliser l'application de programmation robotique NiryoStudio permettant une programmation simple et intuitive en Blockly [6] ou Python. Dans notre étude, voulant cibler un environnement davantage industriel, cet outil ne sera pas exploité.

D'un point de vue caractéristique, le Ned 2 dispose d'une charge utile de 300g, d'une portée de 490mm avec une répétabilité de +/- 0,5mm. Il dispose d'une connectivité WiFi et Ethernet, et peut accueillir en outil pinces, ventouses et électroaimant. Associé à ce robot, nous disposons d'un convoyeur connecté à la baie du robot.

Figure 1. Plateforme d'expérimentation

Le robot est connecté par un câble RJ45 à un PC où est installé le logiciel EAE qui permettra de développer l'application en langage IEC 61499 et d'exécuter cette application en servant de plateforme d'exécution (voir section suivante).

Un serveur Node-RED est exécuté sur un autre PC connecté au même réseau, il nous servira soit de passerelle, soit d'Interface Homme Machine (IHM) dans nos expérimentations. L'avantage de Node-RED est que l'IHM peut être disponible sur n'importe quel équipement connecté au réseau comme un autre PC, une tablette ou un smartphone par exemple. Finalement, l'ensemble des équipements communiquent grâce au protocole Modbus TCP (Figure 2).

[2] https://niryo.com/

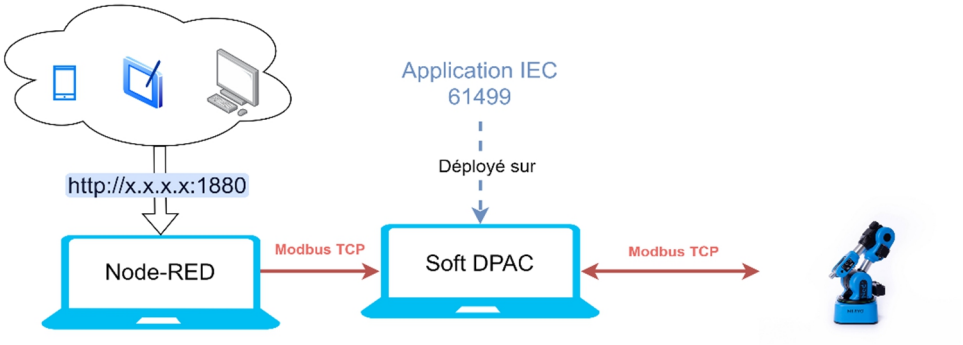

Figure 2. Architectures logiques de la plateforme d'expérimentation

3. Programmation en IEC 61499 avec EAE

3.1. IEC 61499

Le développement et l'exécution d'une application IEC 61499 nécessite un environnement de développement intégré et une plateforme d'exécution sur laquelle elle sera déployée. EcoStruxure Automation Expert (EAE) est l'environnement de développement intégré développé par Schneider Electric. Il permet de développer des applications IEC 61499 et de les déployer sur des automates programmables industriels ou sur des ordinateurs industriels grâce aux Soft dPAC, des plateformes d'exécution Linux et Windows. EAE propose une offre d'automatisation universelle en découplant les parties logicielles et matérielles notamment depuis la création de l'association UniversalAutomation.org[3] en 2021. Cette association regroupe des industriels et des universités dans l'objectif de promouvoir une plateforme d'exécution partagée basée sur la norme IEC 61499. EAE est l'un des outils logiciels permettant de déployer des applications dans cette plateforme d'exécution. Il permet l'interopérabilité et la portabilité des applications entre des contrôleurs de fournisseurs différents.

Une des particularités de l'IEC 61499 est l'exécution des applications de façon événementielle. A la différence des applications programmées cycliquement en IEC 61131, si aucun événement ne se produit en IEC 61499, alors aucune action n'est effectuée ce qui permet de réduire la charge des contrôleurs. L'exécution des applications IEC 61499 étant exclusivement pilotée par événements, les blocs fonctionnels qui constituent les applications sont composés d'événements en entrée et en sortie auxquels sont associés des données.

Ces blocs fonctionnels (Figure 3) sont activés par le déclenchement d'un événement et actualisent les données d'entrées. Ces blocs peuvent être de plusieurs types :

- Les blocs fonctionnels basiques se composent d'une machine à état, le graphe de contrôle d'exécution (ECC) qui fait appel à des algorithmes et déclenche les événements de sorties,

- Les blocs fonctionnels composites encapsulent d'autres blocs fonctionnels,

- Les blocs fonctionnels de services dont le comportement est caché et qui permet de gérer les entrées, sorties et interfaces réseaux.

[3] https://universalautomation.org/fr

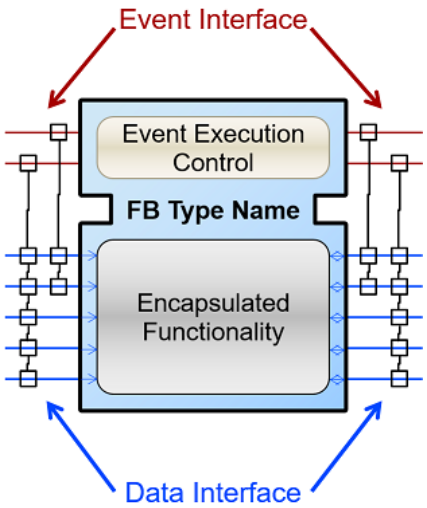

Figure 3. *Structure d'un bloc fonctionnel (https://iec61499.com/)*

3.2. Communication

Le protocole de communication choisi pour faire l'interface entre l'API et le robot Ned2 est le Modbus TCP.. Pour cela, nous utilisons les registres et spécifications nécessaires à l'utilisation du Modbus fournis via le lien : *https://docs.niryo.com/api/modbus/use-the-modbus-tcp-server/*. Les entrées discrètes et les registres d'entrée sont des tables en lecture seule. Ils sont utilisés pour conserver l'état du robot. Les bobines et les registres de maintien sont des tables en lecture/écriture. Ils sont eux utilisés pour donner des commandes au robot.

Les nombres flottants sont tous encodés avec une précision de 32 bits. Une entrée de registre étant de 16bits, tous les nombres flottants sont encodés sur 2 entrées. Par ailleurs, tous les caractères sont codés sur 4 bits. Une entrée de registre étant de 16 bits, chaque adresse contient donc 2 caractères.

Il nous est donc nécessaire d'identifier les variables désirées en lecture et écriture, mais également d'effectuer une conversion sur leur adresse (de type Mid Big Endian).

Pour notre application, nous utiliserons les adresses Modbus TCP suivantes :

Tableau 1. Code fonction Modbus

Code fonction	Fonction	Description
1(0x01)	Read coils	Read output bit
2(0x02)	Read discrete inputs	Read input bit
3(0x03)	Read Holding registers	Read output word
4(0x04)	Read input registers	Read input word
5(0x05)	Write single coil	Write one bit output
6(0x06)	Write single register	Write one word output
15(0x0F)	Write multiple coils	Write a number of output bits
16(0x10)	Write multiple registers	Write a number of outputs words
23(0x17)	Read/Write multiple registers	Read a number of input words / Write a number of outputs words

Tableau 2. Commande Modbus du robot Niryo Ned 2

Composant	Adresse	Type	Nom	Description	FC
Convoyeur	53	bool	Convoyeur 1 associé	True lance un scan de tous les convoyeurs	FC5
	73	bool	Convoyeur 1 en marche	/	FC1
	93	bool	Direction du Convoyeur 1	0 = arrière | 1 = avant	FC5
	87	int	Vitesse du Convoyeur 1	Pourcentage 0-100 démarre lorsque la valeur est > 0	FC6
	100	int	Convoyeur 1 ID	ID du Convoyeur 1	FC4
Capteur de présence	4	bool	Entrée Digitale 5	Contrôle de la valeur de l'entrée numérique 0= LOW | 1 = HIGH	FC2
Commande du bras	115	bool	Calibration nécessaire	/	FC1
	116	Bool	Calibration	Exécuter la Calibration	FC5
	113	bool	Robot en mouvement	/	FC5
	50-51	float	Cible Joint 1	Position de la cible du Joint 1 (rad)	FC6
	52-53	float	Cible Joint 2	Position de la cible du Joint 2 (rad)	FC6
	54-55	float	Cible Joint 3	Position de la cible du Joint 3 (rad)	FC6
	56-57	float	Cible Joint 4	Position de la cible du Joint 4 (rad)	FC6
	58-59	float	Cible Joint 5	Position de la cible du Joint 5 (rad)	FC6
	60-61	float	Cible Joint 6	Position de la cible du Joint61 (rad)	FC6
	74	int	Type de mouvement	0 = Move_joint | 1=Move_pose | 2=Move_linear	FC6
	142	int	Vitesse du bras	Pourcentage	FC6
Pince	51	bool	Activation de l'outil	0 = Relâcher avec l'outil 1 = Prendre avec l'outil	FC5
	107	Int	Vitesse d'ouverture	Pourcentage de vitesse	FC6
	108	Int	Couple maximal d'ouverture	Pourcentage de couple	FC6
	109	int	Couple d'ouverture et de maintien	Pourcentage de couple	FC6
	110	int	Vitesse de fermeture	Pourcentage de vitesse	FC6
	111	int	Couple maximal de fermeture	Pourcentage de couple	FC6
	112	int	Couple de fermeture et de maintien	Pourcentage de couple	FC6

Le tableau 3 ci-dessous indique pour information les angles minimaux et maximaux pour chacune des articulations du bras de robot.

Tableau 3. Limites d'angles des axes du Ned 2

Articulation	Min (rad)	Max (rad)
J1 – Base	-2,99	2,99
J2 – Shoulder	-1,83	0,61
J3 – Elbow	-1,34	1,57
J4 – Forearm Rotation	-2,09	2,09
J5 – Wrist	-1,92	1,92
J6 – Hand Rotation	-2,53	2,53

3.3. Programmation

Dans l'environnement EAE, il convient tout d'abord de configurer un CAT matériel, ici un équipement MODBUS. A partir de ce matériel, il est nécessaire d'ajouter un client Modbus de type Standard.IoModbus.MODBUSGENTCPS. Une ligne MODBUSGENTCPS est créée, et c'est dans les propriétés de ce module qu'il est possible de configurer l'IP et le port du Niryo Ned2. C'est également à partir de cet équipement qu'il est possible d'ajouter les fonctions Modbus pour lire et écrire dans le serveur (Figure 4).

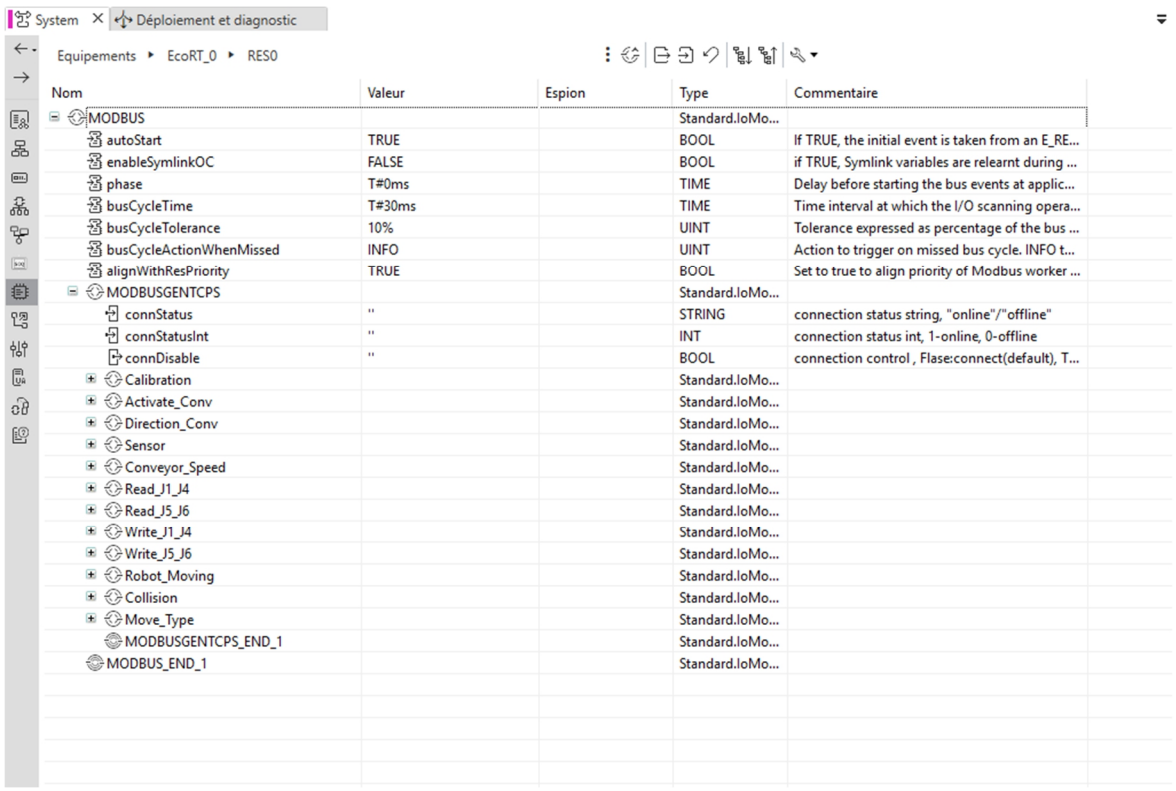

Figure 4. *Configuration du client Modbus sous EAE*

Dans notre étude, nous voulons, (i) calibrer le bras, (ii) piloter chaque axe, et (iii) commander le convoyeur (en activation, vitesse et direction) (Figure 5).

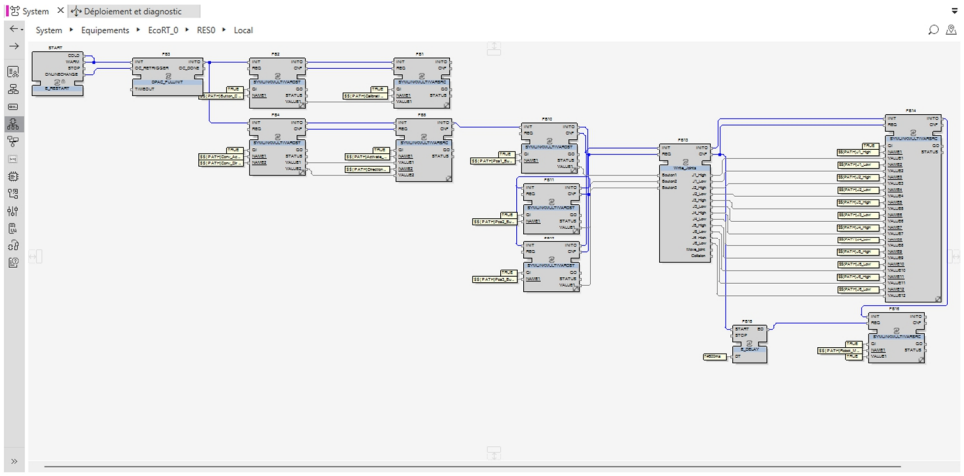

Figure 5. *Vue d'ensemble du programme (détaillée par la suite)*

Nous utilisons également Node-RED comme IHM (Figure 6). Pour cela, la lecture des boutons provenant de l'IHM permettra de lancer la calibration (FB2) et l'Activation et Direction du convoyeur (FB4), mais aussi une écriture de calibration du bras (FB1) et d'activation et direction du convoyeur (FB5) (Figure 7). Les blocs fonctionnels SYMLINKMULTIVARDST et SYMLINKMULTIVARSRC permettent de faire l'interface entre les variables définies dans la configuration Modbus de la Figure 3 et l'application.

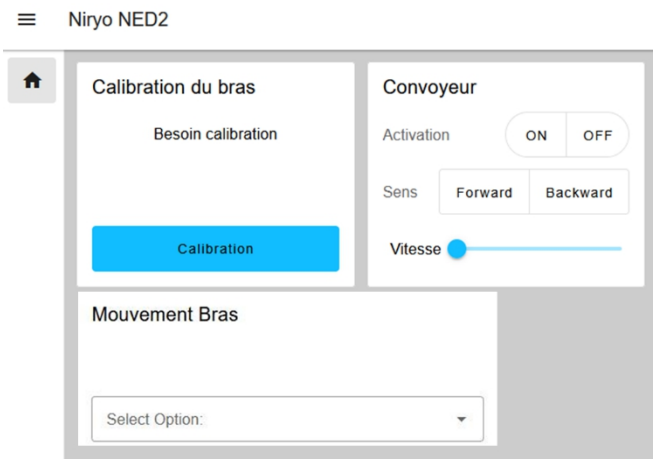

Figure 6. *IHM NodeRed*

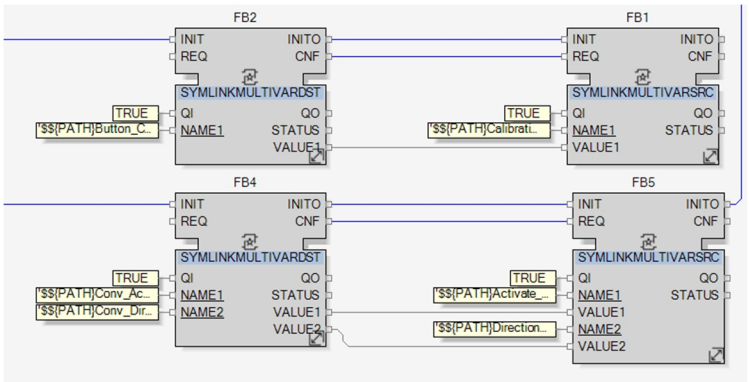

Figure 7. *FB d'IHM*

Concernant les mouvements du bras (Figure 8), les blocs FB10, 11 et 12 permettent la lecture d'état de 3 boutons (position 1, position 2 et position 3) provenant de l'IHM. Le bloc FB13 est un bloc personnalisé par un code ST. Ce dernier permet selon la position voulue d'envoyer les bonnes coordonnées des 6 joints au bloc FB14 qui a pour fonction d'écrire dans les registres associés. A l'initialisation, le bloc FB13 envoie 0 à l'ensemble des joints. Enfin, le bloc FB16 permet d'envoyer la commande de mouvement du bras. En effet, pour commander le bras robotique au travers de son serveur Modbus TCP, il faut écrire la valeur des joints dans les registres correspondants puis mettre à 1 la bobine à l'adresse 113 pour déclencher le mouvement.

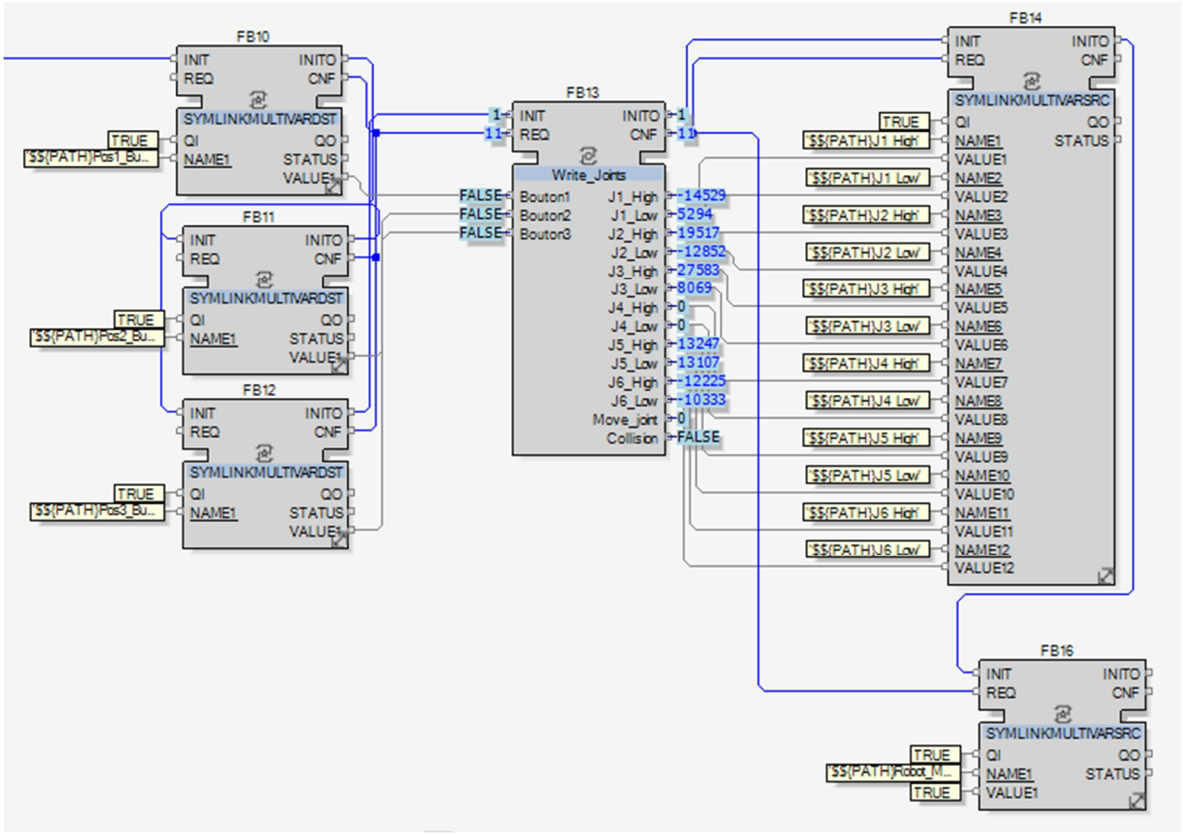

Figure 8. *Commande de mouvements du bras*

4. Problématique rencontrée

Lors de cette expérimentation, nous nous sommes confrontés à divers problématiques.
Premièrement, l'implémentation du serveur Modbus TCP dans la Raspberry Pi qui contrôle le robot ne permet d'effectuer qu'une action (lecture ou écriture) à la fois. C'est-à-dire que lorsqu'une action de déplacement ou de calibration est envoyée comme indiqué dans la section précédente, il n'est pas possible de lire dans le serveur ou d'envoyer d'autres commandes tant que le déplacement n'est pas fini. Cela entraînait également des déconnexions au niveau du client Modbus d'EAE. Il est alors compliqué de commander le Niryo Ned2 avec un API s'il n'est pas possible d'avoir les informations des capteurs en temps-réel.
L'autre problématique rencontrée se trouve dans la lecture ou l'écriture de deux registres de 16 bits pour envoyer une valeur de 32 bits de type Float en format Mid Big Endian. Chacune des opérations nécessite un bloc fonctionnel pour convertir les valeurs dans un sens ou dans l'autre.

La figure 9 montre l'algorithme en ST qui permet à partir des valeurs de deux registres d'obtenir la valeur de type Float correspondant à la valeur du joint. Il est nécessaire d'effectuer plusieurs conversions suivies de décalages de bit pour réaliser cette opération.

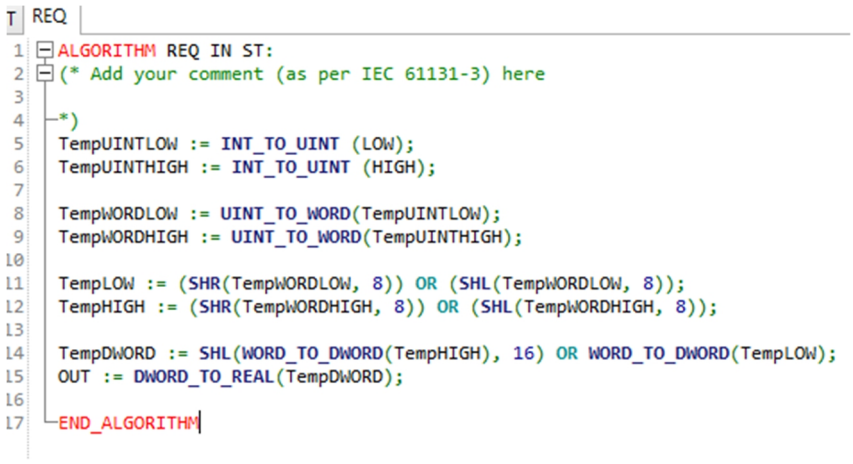

Figure 9. *Programme de conversion pour récupérer la valeur du joint*

5. Conclusion et perspectives

Ce papier illustre la programmation d'un robot académique via une plateforme d'automatisme IEC 61499. Il y est décrit l'environnement du robot Ned2 mais surtout la complexité de communication et à moindre mesure de programmation en IEC 61499 sous EAE.

La continuité du projet se traduit notamment par un rapprochement avec la société Niryo qui, suite à nos tests, développe certains correctifs sur leur robot (notamment autour de Modbus).

Pour le moment, aucune expérimentation auprès des étudiants n'a été réalisée. Celle-ci est en préparation avec une observation de séance ainsi qu'un questionnaire de fin de module. Les compétences attendues sont les capacités d'adaptation de l'environnement, mais aussi la compréhension de l'évolution événementielle de la norme IEC 61499.

Par ailleurs, nous souhaitons développer également l'application d'automatisme en programmant le Ned2 sous d'autres environnement comme TIA Portal de Siemens (via PLCSim et API réel) mais également certain avec les API Schneider Electric. Par ailleurs, d'autres fonctionnalités. Nous souhaitons également intégrer un serveur OPCUA dans le Niryo afin de faciliter la communication et l'interopérabilité du produit.

Enfin, point de vue EAE, d'autres cobots vont être à l'étude (Universal Robot) mais également des applicatifs matériels spécifiques comme des caméras événementielles.

6. Remerciements

Ce travail a bénéficié en partie d'une aide de financement de l'Appel à Manifestation d'Intérêt (AMI) du Réseau d'ESR du site Champardennais.

Références

[1] A. Gasparetto and L. Scalera, *A Brief History of Industrial Robotics in the 20th Century*, Advances in Historical Studies, vol. 08, n°01, (2019), pp. 24–35.

[2] A. Parant, L. Arcese, S. Martinez-Martinez and A. Marguery, *Cooperation and synchronization of robotic tasks using a digital twin*, *21st International Conference on Informatics in Control, Automation and Robotics*, vol. 2, pp 73-79, Porto, Portugal, (ICINCO'24), 2024.

[3] M. Golovianko, V. Terziyan, V. Branytskyi and D. Malyk, *Industry 4.0 vs. Industry 5.0: Co-existence, Transition, or a Hybrid*, Procedia Computer Science, vol. 217, pp. 102-113, (2023).

[4] W.W. Dai and V. Vyatkin, *A case study on migration from IEC 61131 PLC to IEC 61499 function block control*, *7th IEEE International Conference on Industrial Informatics*, pp. 79-84, 2009.

[5] J.H. Christensen, T. Strasser, A. Valentini, V. Vyatkin, A. Zoitl, J. Chouinard, H. Mayer and A. Kopitar, *The IEC 61499 function block standard : Software tools and runtime platforms*. ISA Automation Week, (2012).

[6] D. Weintrop, D. C. Shepherd, P. Francis and D. Franklin, *Blockly goes to work: Block-based programming for industrial robots*, IEEE Blocks and Beyond Workshop (B&B), Raleigh, NC, USA, (2017), pp. 29-36.

Biographie des auteurs

Alexandre Philippot : Après un DEA en Optimisation et Sûreté des Systèmes à l'Université de Reims Champagne-Ardenne en France, Alexandre Philippot a obtenu son doctorat en Diagnostic des Systèmes à événements discrets à l'Université de Reims Champagne-Ardenne en France (URCA). Il a effectué un PostDoc à l'École Normale Supérieure de Cachan (ENS-Cachan) en France (2007) avant d'être en poste de Maître de conférences à l'URCA. Il a défendu son HDR en 2018 et est Professeur des Universités à l'URCA depuis 2023. Thèmes de recherche : Commande, Reconfiguration et Diagnostic des Systèmes à événements discrets.

Stéphane Lecasse : Stéphane Lecasse a obtenu le diplôme de Brevet de Technicien Supérieur en électronique à Reims en 1999. Depuis 2001, il fait partie de l'équipe technique du Centre de Recherche en Sciences et Technologies de l'Information et de la Communication (CReSTIC) de l'Université de Reims Champagne-Ardenne. Ses travaux portent principalement sur l'appui technique dans le domaine de l'industrie 4.0 (jumeaux numériques, virtual commissioning, objets connectés…)

Alexandre Parant : Alexandre Parant a obtenu le diplôme de Master en informatiques industrielles de l'Université de Reims Champagne-Ardenne (URCA) en 2019, puis le doctorat en automatique et traitement du signal de l'URCA/CReSTIC en 2023. Depuis 2024, il est Enseignant-Chercheur au Laboratoire d'Innovation Numérique pour les Entreprises et les Apprentissages au service de la Compétitivité des Territoires (LINEACT) de CESI. Ses travaux de recherches portent principalement sur la robotique, la conception de commande et le pilotage des systèmes cyber-physiques de production, notamment à l'aide des jumeaux numériques.

Laurent Arcese : Laurent Arcese a obtenu le diplôme de Master en automatique de l'Université de Lyon 1 en 2008, puis le doctorat en automatique et traitement du signal de l'Université d'Orléans/INSA CVL en 2011. Depuis 2012, il est Maître de conférences au Centre de Recherche en Sciences et Technologies de l'Information et de la Communication (CReSTIC) de l'Université de Reims Champagne-Ardenne. Ses travaux de recherche portent principalement sur la synthèse de lois de commande à partir de fonctions de Lyapunov, ainsi que sur le développement d'observateurs pour différentes classes de systèmes non linéaires, appliqués sur des systèmes cyber-physiques variés (systèmes robotisés, drones, micro/nanorobotique, ...).

François Gellot : François Gellot est titulaire d'un doctorat en automatique de l'Université de Reims Champagne-Ardenne (URCA). Il est Maître de conférences en automatique à l'Université de Reims Champagne-Ardenne (URCA) et chercheur au CReSTIC (Centre de recherche en sciences et technologies de l'information et de la communication). Ses recherches portent sur la modélisation, l'analyse, la synthèse et la validation des systèmes à événements discrets.