Programmation ST/LD/SFC sous OpenPLC

Culture Sciences de l'Ingénieur

Jean-Philippe ILARY

Édité le 15/03/2021

école normale supérieure paris—saclay

Cette ressource est issue d'une publication du numéro 103 de La Revue 3EI de janvier 2021.
Jean-Philippe Ilary est enseignant au département GEII de l'IUT Ville d'Avray.

Pendant le premier confinement, l'enseignement à distance du module automatisme nous ont amené à mettre en place des solutions pour palier certaines difficultés matérielles et logicielles. Cette matière s'appuyait, en situation normale, sur l'emploi du logiciel Unity de Schneider associé à des Modicon M340. Malgré la mise à disposition, durant cette période de distanciel, d'une solution logicielle orientée simulation, certains étudiants ont eu des difficultés d'ordre matériel (ordinateur Apple, version de Windows etc.), c'est pourquoi nous avons orienté ces étudiants vers la solution logicielle OpenPLC. OpenPLC Editor est une alternative logicielle Open Source qui répond à la norme CEI 61-131. Ce logiciel permet de développer des programmes API sur des plateformes Raspberry/Windows/Linux.

Cette ressource a pour ambition de fournir à un étudiant les bases pour débuter en autonomie avec le logiciel OpenPLC.

1 – Introduction

Après avoir présenté les possibilités de la suite logicielle OpenPLC, ce document, adapté pour cette publication, présente les démarches à suivre par l'étudiant afin qu'il puisse développer en autonomie.

La première partie permet la prise en main du logiciel OpenPLC Editor :

- Configurer le projet,
- Créer un programme LADDER simple et programmer un grafcet en SFC,
- Utiliser le simulateur pour valider le fonctionnement du programme.

La seconde partie permet de valider les programmes du TD programmation LD/ST/SFC :

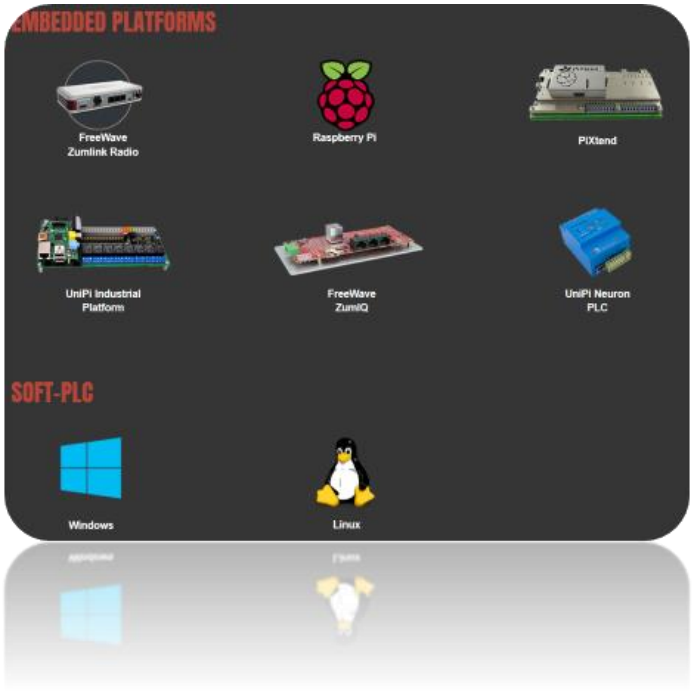
- Programmer un cahier des charges simple en langage LD/ST/SFC,
- Analyser les éventuelles erreurs de compilation,
- Valider les programmes en simulation,
- Rédiger un compte-rendu.

2 – Installation & prise en main

Le projet OpenPLC propose en ensemble d'éléments afin de développer des automatismes en Open source. Il fournit trois parties : un Runtime, un Editeur et un outil de développement d'IHM. L'éditeur est le logiciel qui s'exécute sur votre ordinateur et est utilisé pour créer vos programmes PLC. Enfin, ScadaBR est le constructeur HMI. Avec ScadaBR, vous pouvez créer de belles animations web qui refléteront l'état de votre processus. ScadaBR communique avec OpenPLC Runtime sur Modbus/TCP.

2.1 - L'Automate simulé (Runtime)

Le Runtime doit être installé sur votre appareil (il est responsable de l'exécution de votre programme PLC). Il permet de disposer d'un API à faible coût. Pour l'installer, il suffit de se rendre sur le site https://www.openplcproject.com/runtime et de choisir la version souhaitée, soit pour carte Raspberry (ou équivalent), soit pour Windows, Linux. Exécutez, cela va permettre d'installer les logiciels nécessaires à la simulation d'un API. Des consoles DOS s'ouvriront pour télécharger les logiciels manquants.



L'installation prendra un certain temps !

Il est aussi possible de disposer d'Entrées/Sorties déportées à travers différents matériels comme une carte Arduino ou un ESP8266.

2.2 - L'éditeur de programme

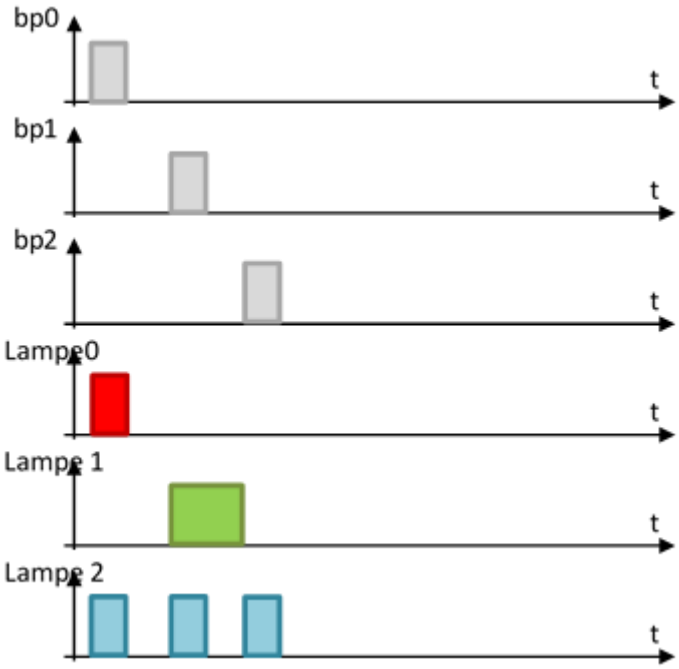
Il faut se rendre sur le site https://www.openplcproject.com/plcopen-editor et choisir la version à télécharger. Pour Mac OS/X voir https://openplc.discussion.community/post/openplc-editor-on-mac-os-x-9905213.

Pour la suite de la ressource, nous travaillerons avec un API sur un système Windows.

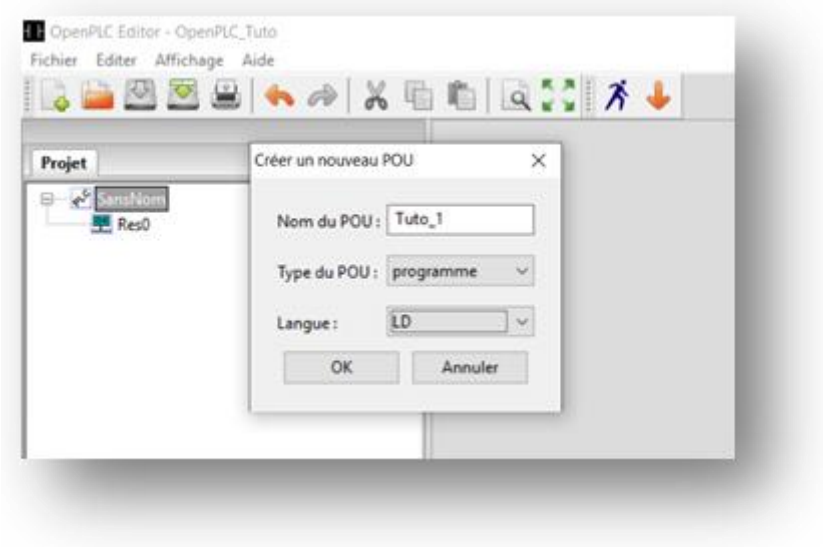
3 – Programmation en LADDER

3.1 - Créer un projet et définir une architecture matérielle

Nous allons créer un programme répondant au chronogramme ci-après. Nous disposerons de trois boutons poussoirs (bp0, bp1 et bp2) et de trois lampes (Lampe0, Lampe1 et Lampe2). Ce sera notre « Bonjour le monde » que tout programmeur connait.



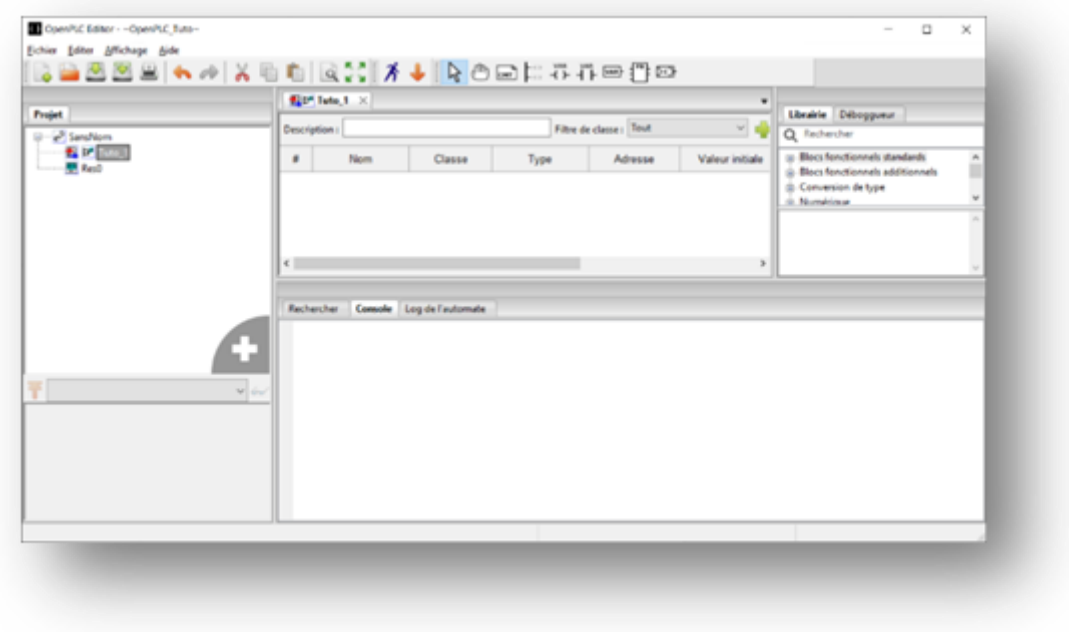
Créer un nouveau projet (CTRL+N) et choisir le langage Ladder (LD)



Sauvegarder le projet dans un répertoire vide !

(Remarque : attention, l'emplacement de sauvegarde peut générer des erreurs lors de la compilation, il suffit alors de le changer)

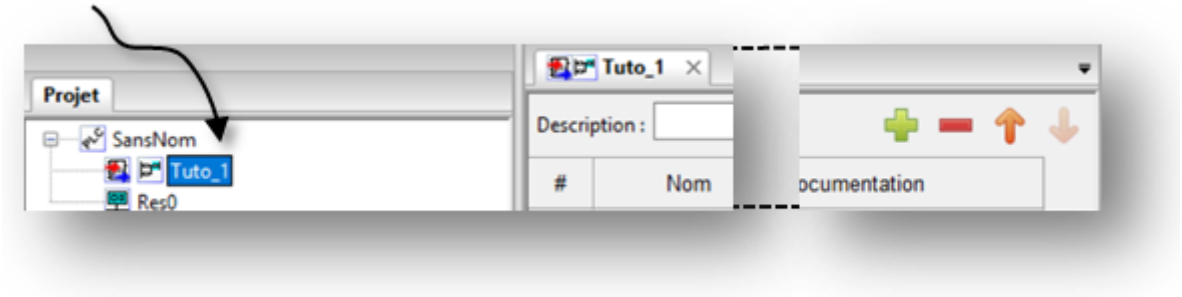
On obtient la fenêtre d'édition suivante qui va nous permettre d'écrire nos programmes API.



3.2 – Associer un symbole aux variables (E/S TOR)

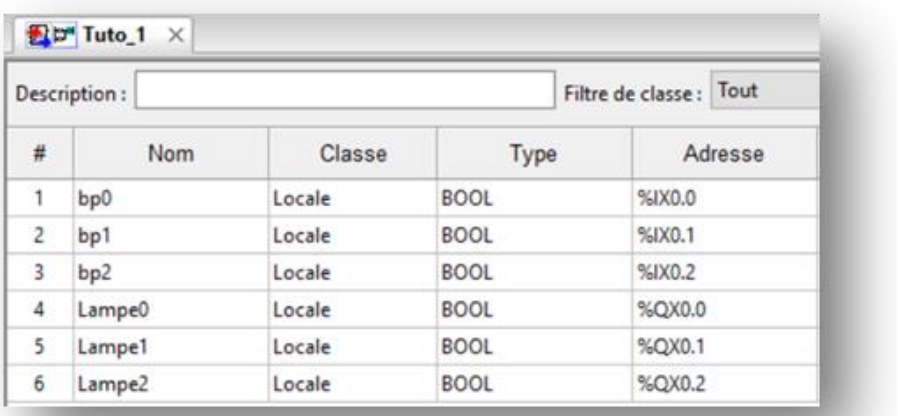
Lors d'une programmation API, il est toujours préférable, comme en programmation informatique classique, de commencer par réfléchir à la structuration du programme, et en automatisme, à réaliser une table d'adressage avec les mnémoniques des variables associées à leur type.

À partir du navigateur projet, double cliquer sur Tuto_1 afin de pouvoir afficher la fenêtre d'édition.



Cliquer sur l'icône afin de rajouter une variable. *(Explication de l'Adressage en Annexe)*

Définir trois variables d'entrée TOR et trois variables de sortie TOR en suivant la table d'adressage ci-après. Les données sont définies en BOOL.

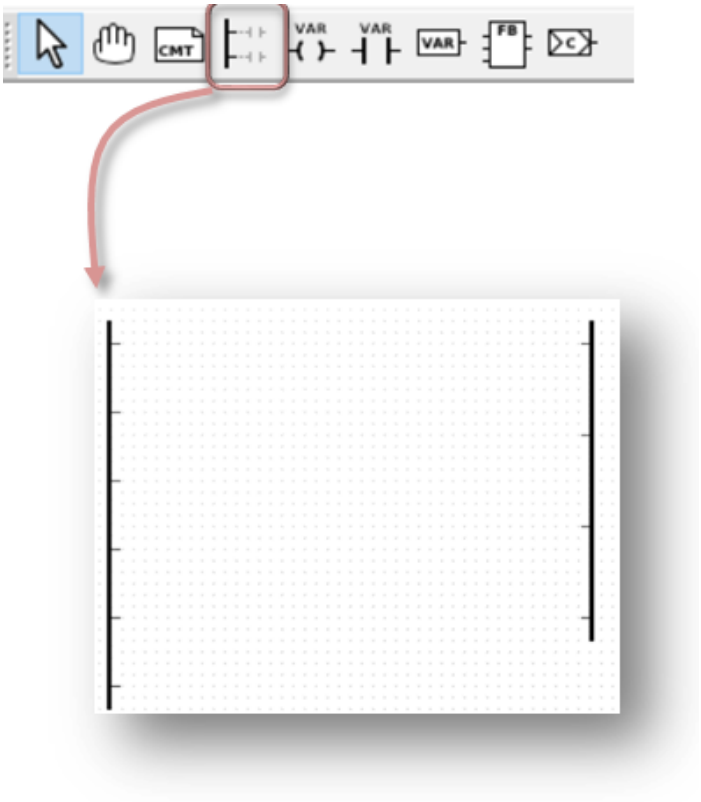


#	Nom	Classe	Type	Adresse
1	bp0	Locale	BOOL	%IX0.0
2	bp1	Locale	BOOL	%IX0.1
3	bp2	Locale	BOOL	%IX0.2
4	Lampe0	Locale	BOOL	%QX0.0
5	Lampe1	Locale	BOOL	%QX0.1
6	Lampe2	Locale	BOOL	%QX0.2

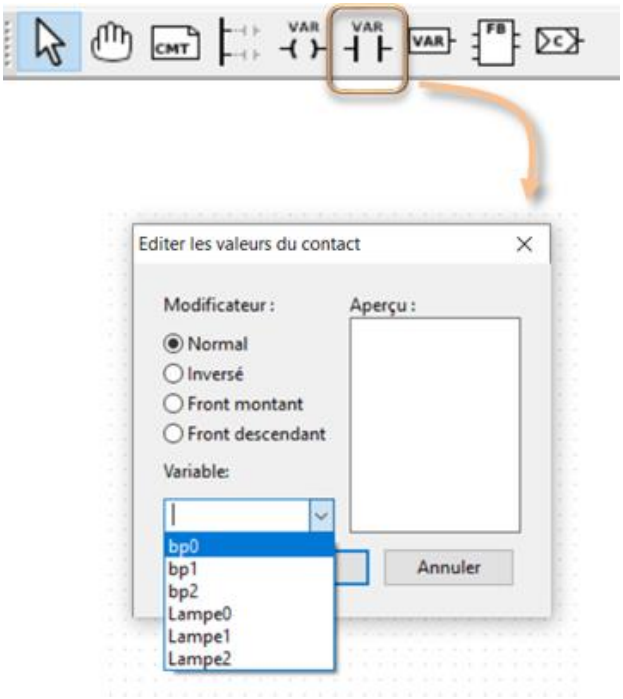
3.3 – Créer un programme en LADDER

La zone de saisie du Ladder est sous le tableau des variables. Cela se fait grâce à la barre d'outils, saisir le programme suivant :

Commencez par placer les deux connexions verticales correspondant à l'alimentation du circuit. Côté gauche 6 départs et côté droit 4 départs. Pour les contacts ou bobines, faire de même.

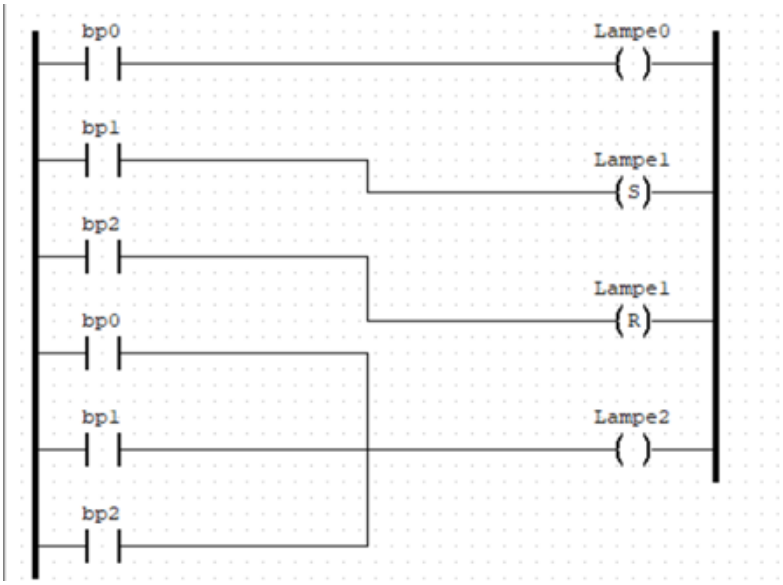


Ensuite, pour les contacts (même démarche pour les bobines) saisir ses caractéristiques.



Ces outils vont permettre d'élaborer le programme API répondant au cahier des charges.

Cela donne :



Il suffit alors d'enregistrer le projet  .

3.4 - Tester et valider le programme

3.4.1 - Compiler

Pour compiler le projet, sélectionner l'icône 

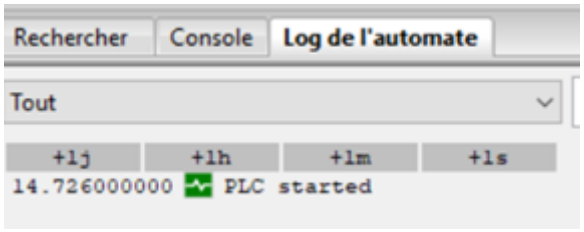
Le compilateur indique en bas de page les éventuels avertissements (non bloquant, à lire quand même) et erreurs (bloquant).

Il vous demande un nom de programme se terminant par .st, car quelque-soit le langage de programmation sélectionné, votre programme est traduit en langage ST. Vous pouvez ouvrir ce fichier avec le blocNote pour vous en rendre compte.

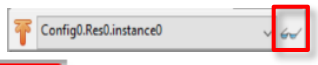
3.4.2 - Se connecter à l'API simulé, exécuter le programme et le valider

Pour se connecter à l'API simulé, sélectionner l'icône : 

Le logiciel compile votre programme, le transférer au simulateur et bascule l'API en RUN. Vérifiez alors que l'API (PLC) est bien en RUN (started).



Pour simuler votre application, cliquer sur Déboguer l'instance dans le panneau gauche en cliquant sur la lunette à droite du texte Config0.Res0.instance0 :



Un nouvel onglet apparait. :

Les lignes en vert sont activées, celles en noir ne le sont pas.

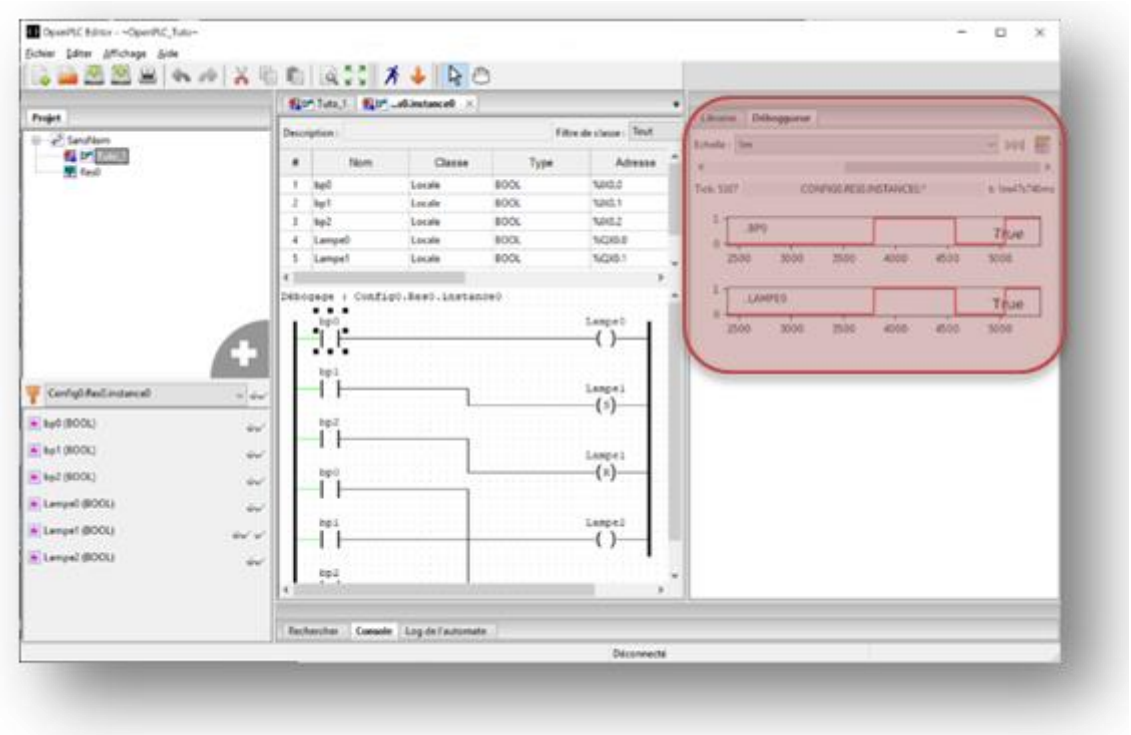
Pour revenir à l'édition du programme, sélectionnez l'onglet p0 sur la capture ci-dessus (pour vous l'onglet portera le nom de votre programme.

Tester le fonctionnement du programme avec la table d'animation et le programme LADDER animé :

- Lampe0 active si bp0 appuyé,
- Lampe1 active si appui sur bp1, l'ordre est mémorisé. Lampe1 est désactivée par l'appui sur bp2,
- Lampe2 active si bp0 appuyé ou bp1 appuyé ou bp3 appuyé.

Lorsque vous avez terminé, déforcez les variables, passer l'automate en STOP (icône 

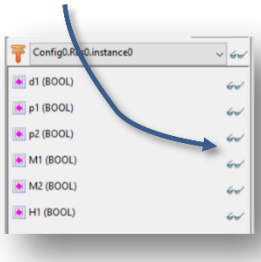
) et revenez sur la fenêtre projet.



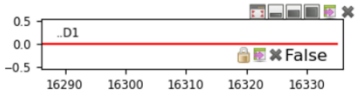
AIDE

Pour visualiser une variable dans l'onglet Déboggueur !

Sélectionner les variables que vous souhaitez voir apparaître dans le chronogramme du déboggueur en cliquant sur les lunettes.

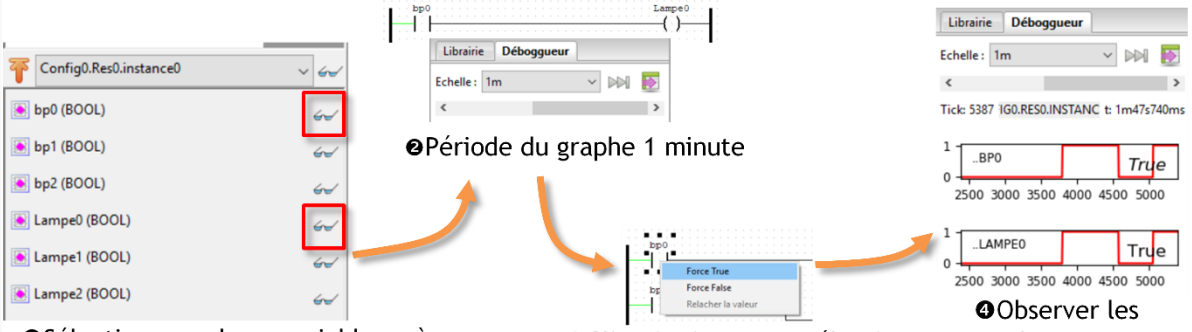


Double cliquer sur la variable dans la fenêtre « Déboggueur » afin de faire apparaître son chronogramme, si celui-ci n'apparaît pas.



Survoler avec la souris, la valeur de la variable (ici FALSE) afin de faire apparaître False . Il faut ensuite cliquer sur le cadenas, la fenêtre suivante apparaît :

Pour observer l'évolution des variables de sorties en fonction de l'état des variables d'entrées en LADDER, il est possible de suivre la démarche ci-après pour valider, par exemple, cette ligne de code :



❶Sélectionner les variables à visualiser

❷Période du graphe 1 minute

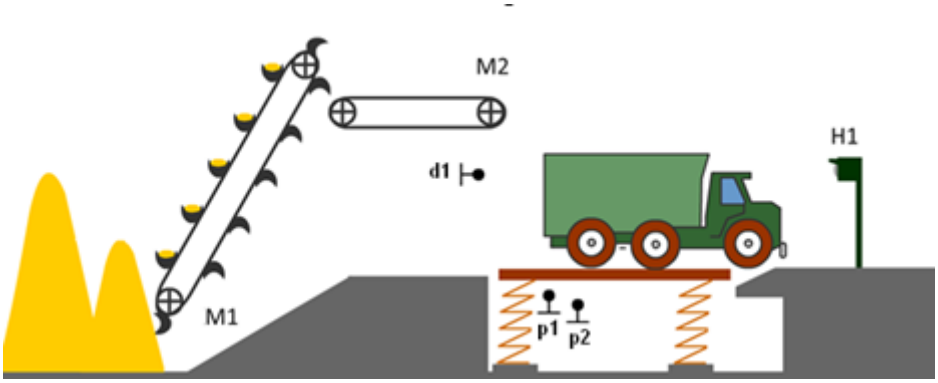
❸Clic Droit pour sélectionner l'état souhaité de la variable

❹Observer les chronogrammes

4 – Programmer en SFC (grafcet)

Afin de comprendre comment procéder pour réaliser un programme SFC, nous nous appuierons sur le cahier des charges suivant :

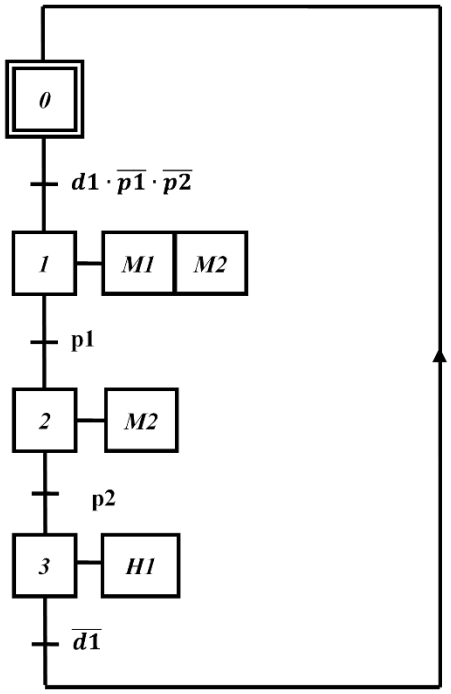
Un camion vide arrive en reculant, il actionne le détecteur de présence d1 ce qui entraîne le démarrage du tapis 1 (moteur M1) et du tapis 2 (moteur M2).



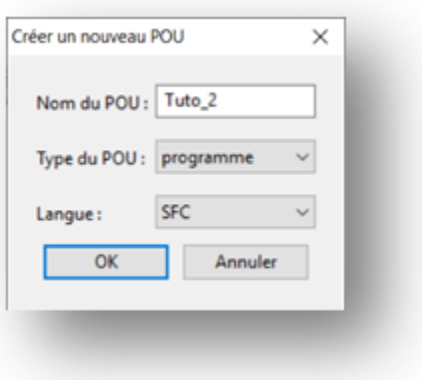
Le camion est placé sur un système de pesée (en contact avec d1). Quand le camion est en partie chargé, l'action de la plateforme sur le détecteur p1 doit stopper le tapis 1, et maintenir le tapis 2 en action.

Le tapis 2 continue de fonctionner jusqu'à l'activation du détecteur p2. Le voyant H1 s'allume pour signaler que le chargement est terminé.

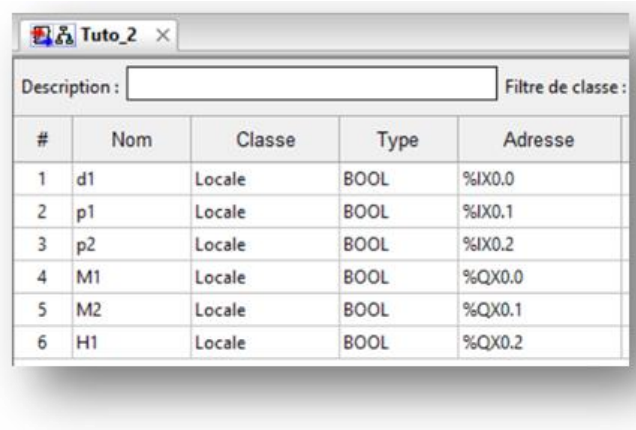
Le grafcet suivant répond au cahier des charges :



Commencer par créer un nouveau projet, le nommer Tuto_2. La programmation se fera en SFC.



Ensuite, définir les symboles qui seront utilisés dans le programme. Il faut définir le type et l'adresse pour chacune des variables.

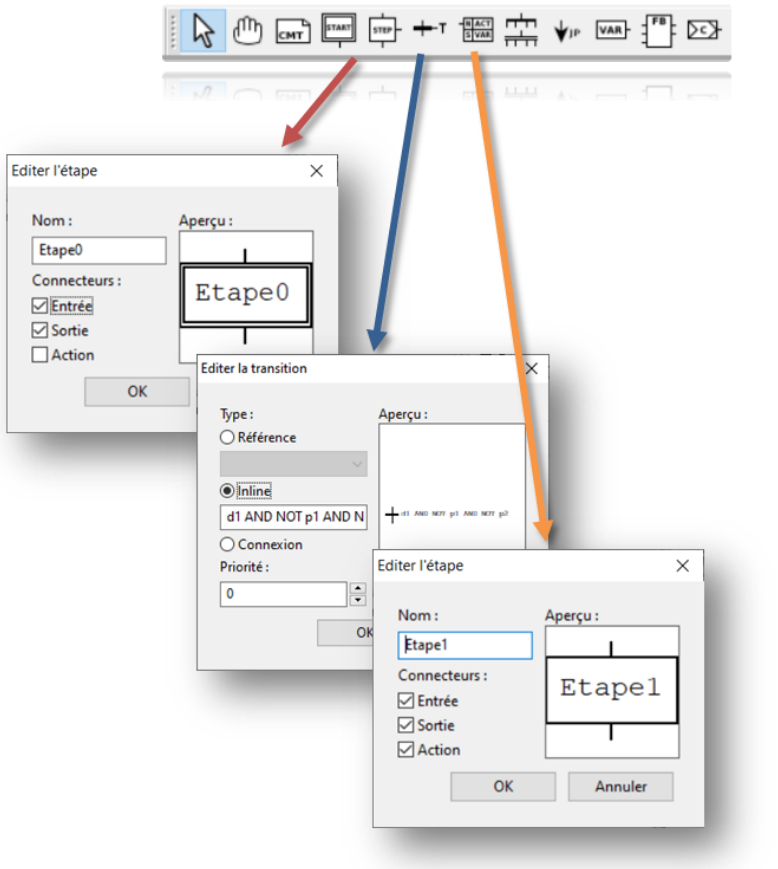


#	Nom	Classe	Type	Adresse
1	d1	Locale	BOOL	%IX0.0
2	p1	Locale	BOOL	%IX0.1
3	p2	Locale	BOOL	%IX0.2
4	M1	Locale	BOOL	%QX0.0
5	M2	Locale	BOOL	%QX0.1
6	H1	Locale	BOOL	%QX0.2

Il ne reste plus qu'à dessiner le grafcet !

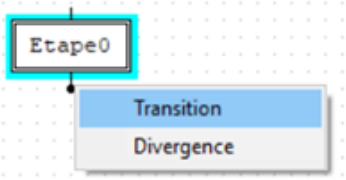
Palette d'outils

Utilisez les icônes disponibles pour créer le grafcet demandé :



Mise en place des transitions et réceptivités

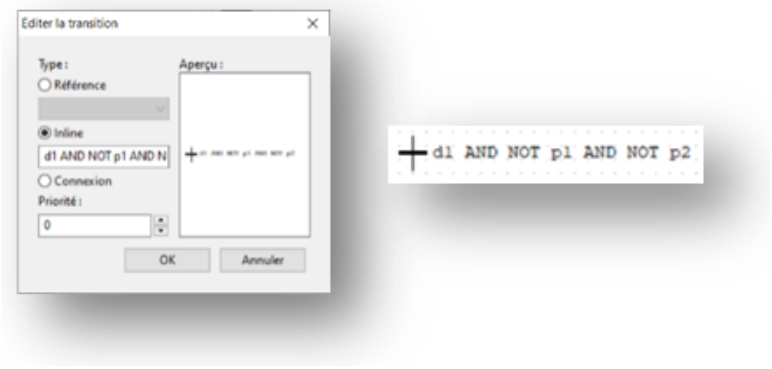
Pour réaliser la construction du squelette du grafcet, il est souhaitable de suivre la démarche ci-dessous :



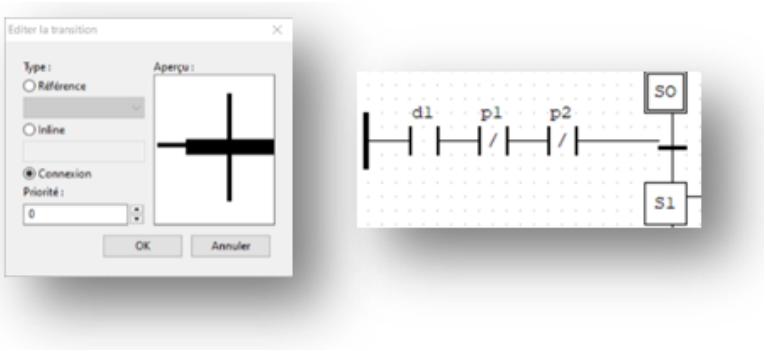
Clic Gauche sur l'extrémité sous l'étape, et vous obtiendrez le menu proposant Transition/Divergence. En choisissant « Transition », la fenêtre « Editer la transition » apparaît.

Il est alors possible de saisir la réceptivité de trois façons, on verra ici deux solutions.

En choisissant ⊙ **Inline**, il sera possible de saisir l'équation logique de la réceptivité suivant la norme CEI 61-131 avec les opérateurs logiques AND, OR, XOR, NOT :

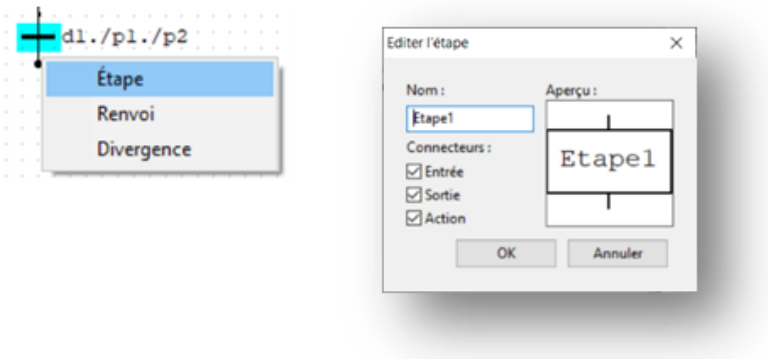


En choisissant ⊙ **Connexion**, il sera possible d'écrire l'équation logique de la réceptivité en langage ladder en commençant à gauche par une barre d'alimentation et en rebouclant sur la transition.

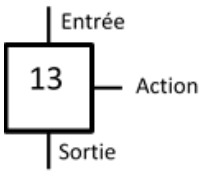


On procédera de même pour la réalisation d'une étape supplémentaire :

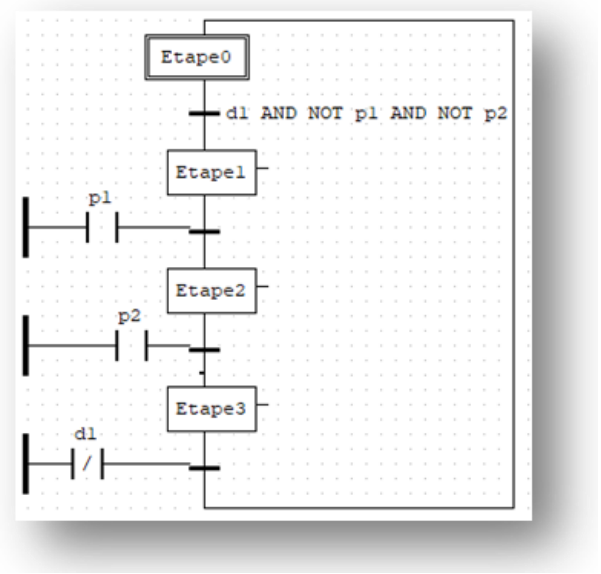
Cette sélection fait apparaître la fenêtre d'édition de l'étape.



Remarque : Une étape peut avoir 3 connexions (Connecteur)

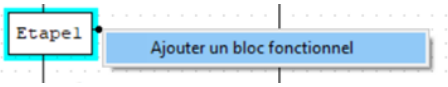


Vous devez obtenir par exemple :

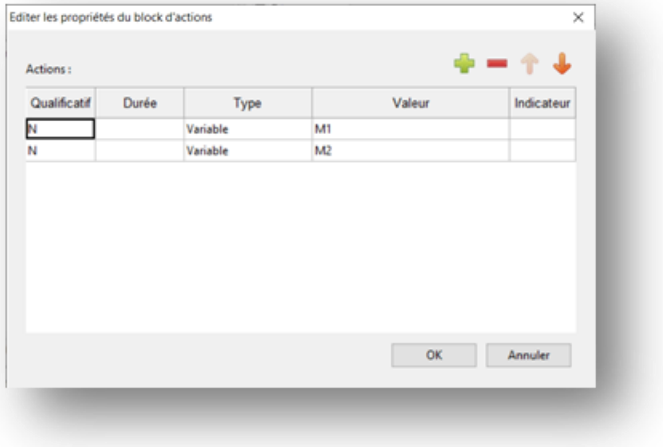


<u>Réalisation des Actions</u>

Ensuite, il faut affecter les actions aux étapes. Pour cela, cliquez sur le départ de l'action. Le menu « Ajouter un bloc fonctionnel » apparaît alors :



Apparaît alors la fenêtre ci-dessous. Les qualificatifs sont ceux définis par la norme, comme indiqué ci-après.



Comme on peut voir ci-dessous, le qualificatif N permet que l'action soit active tant que l'étape l'est.

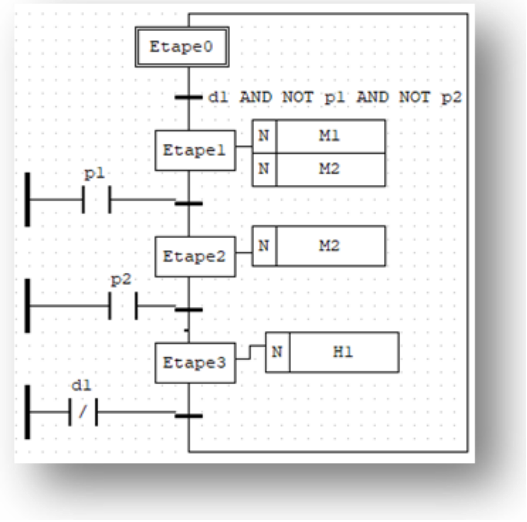
Qualificatif

Pour associer des actions à des étapes CEI, vous disposez des qualificatifs (qualifier) suivant:

N	Non-stored (non mémorisé)	l'action demeure active aussi longtemps que l'étape demeure active
R	overriding Reset (remise à zéro prioritaire)	désactivation de l'action
S	Set (Stored) (positionné (mémorisé))	activation de l'action, qui demeure ensuite active jusqu'au prochain reset
L	time Limited (limite dans le temps)	activation de l'action pendant une durée déterminée
D	time Delayed (temporisé)	activation de l'action après un certain temps, pour autant que l'étape demeure active
P	Pulse (impulsion)	l'action est exécutée exactement une fois, lorsque l'étape est activée
SD	Stored and time Delayed (mémorisé et temporisé)	activation de l'action après un certain temps; l'action demeure ensuite active jusqu'au prochain reset
DS	Delayed and Stored (temporisé et mémorisé)	activation de l'action après un certain temps, pour autant que l'étape demeure active; l'action demeure alors active jusqu'au prochain reset
SL	Stored and time Limited (mémorisé et limité dans le temps)	activation de l'action pendant une durée déterminée

Les qualificatifs L, D, SD, DS et SL doivent être accompagnés d'une spécification de temps sous format de constante TIME, p.ex. L T#5s.

On doit obtenir au final une écriture équivalente à :



La programmation du grafcet est terminée.

Rappel : Pour revenir à l'édition du programme, appuyer sur Stop et sélectionner l'onglet du programme :

En programmeur, il est de notre devoir de commenter le programme. Pour ajouter des commentaires sur la page Chart, utiliser le raccourci en pressant la touche F8.

5 – Validation du fonctionnement

Maintenant que le programme est complet, on va commencer par le compiler (rappel : le dossier du projet doit être dans un répertoire bien accessible (pas sur un lecteur réseau par exemple). Zéro erreur, vous pouvez alors l'exécuter .

Ensuite, comme pour la programmation LADDER, il est possible de superviser les variables (voir Aide plus haut). Cela se fait par et le résultat apparaît à droite dans la fenêtre Déboggueur.

Si vous avez un API compatible OpenPLC (Raspberry par exemple), il vous est possible d'y transférer le programme et de l'exécuter, voir le tutorial en ligne :
https://www.openplcproject.com/reference/basics/upload

6 – Création d'une supervision

Il s'agit ici d'un contrôle de surveillance et d'acquisition de données (type SCADA) qui vous permet de créer des écrans interactifs, également appelés IHM pour une supervision locale), pour vos projets d'automatisation. ScadaBR peut interagir avec plusieurs PLCs, y compris un API OpenPLC.

La façon la plus simple d'installer ScadaBR est d'utiliser une machine virtuelle. Dans la machine virtuelle, tout ce que vous avez à faire est de charger le fichier d'image ScadaBR dans VirtualBox par exemple pour avoir un environnement ScadaBR prêt à l'emploi sur votre système (voir le site pour le détail de l'installation : https://www.openplcproject.com/reference/scadabr/)

7 – Conclusion

Ce travail n'est pas terminé, mais vu les circonstances, il m'a semblé intéressant de le partager en l'état. Les sept pages d'annexes sont des outils qui serviront aux étudiants, à qui ce tutoriel est destiné.
Je remercie, pour leur aide Claire BASSET (IUT GEII - Ville d'Avray) et Jean-Michel GAY (BTS ET - Versailles).

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay