OPENPLC – TRANSFORMER UN RASPBERRY EN API

JEAN-PHILIPPE ILARY
Département GEII – IUT Ville d'Avray
50 Rue de Sèvres, 92410 Ville d'Avray
jpilary@parisnanterre.fr

Résumé : Dans le numéro précédent de la revue (n°103), j'ai présenté l'éditeur de programme du projet OpenPLC. Maintenant que nous savons écrire un programme, il faut une plateforme pour l'accueillir et l'exécuter. Il faut aussi que le monde extérieur puisse interagir avec cet automate, soit directement, soit à l'aide d'entrées/sorties déportées ou communiquer sur le Web.
Comme le premier article, j'essaye de faire en sorte qu'un étudiant puisse utiliser directement le document pour mettre en œuvre en toute autonomie, ce nouvel API.

I/ Introduction

Après avoir présenté les possibilités du projet OpenPLC, ce document s'attachera à décrire les démarches à suivre par l'étudiant afin qu'il puisse mettre en place l'API en autonomie et réaliser un programme de test.

Tout d'abord, une présentation des outils proposés par le projet OpenPLC (si vous n'avez pas le premier article paru dans le n°103).

La première partie permet à l'étudiant la mise en place des logiciels nécessaires au fonctionnement de Raspberry en mode API

La seconde partie applique la même méthode, mais pour l'Arduino, qui devient ainsi des entrées/sorties supplémentaires.

La troisième partie permet de mettre en place un premier programme de test avec le Raspberry uniquement.

En quatrième partie, on rajoutera l'Arduino, qui complète les E/S TOR du Raspberry par des entrées analogiques et des sorties PWM (facilement convertible en analogique).

Enfin, on abordera la communication Modbus/TCP entre le Raspberry/API et un programme Labview.

II/ Le projet OpenPLC

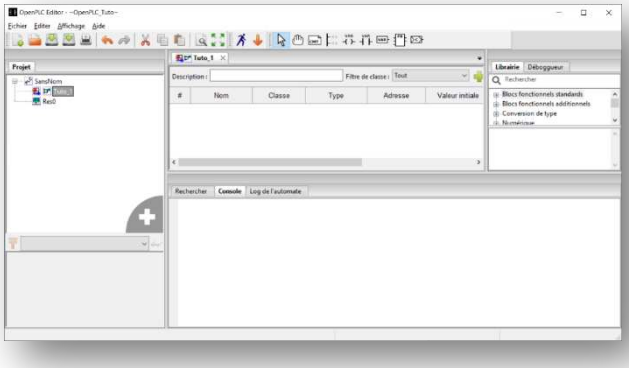
Le projet OpenPLC propose en ensemble d'éléments afin de développer des automatismes en Open source. Il fournit trois parties : un Runtime, un Editeur et un outil de développement d'IHM. Le Runtime fait l'objet de cet article. L'éditeur est le logiciel qui s'exécute sur votre ordinateur et est utilisé pour créer vos programmes PLC (article précédent). 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.

II.1/ L'éditeur de programmes

Il faut se rendre sur le site du projet 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 l'article, nous travaillerons avec un éditeur sur un système Windows.

L'écran prend l'aspect suivant :



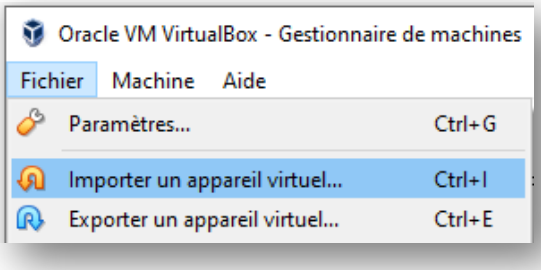
Quelque soit le langage de programmation choisi, le compilateur traduit le programme en langage ST. Il est intéressant de comparer le code généré avec le programme de départ des étudiants.

II.2/ Développement d'une IHM

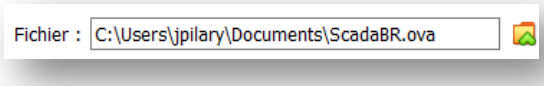
L'outil proposé sur le site est très bien fait. C'est une machine virtuelle que l'on récupère sur le site à l'adresse :
https://www.openplcproject.com/reference/scadabr/

La démarche (que l'on retrouve sur le lien ci-dessus) qu'il faut suivre est la suivante :

1. Téléchargez VirtualBox à partir du site officiel,
2. Téléchargez le fichier d'images virtuelles ScadaBR,
3. Installer VirtualBox en exécutant l'installateur téléchargé à l'étape 1.
4. Exécutez VirtualBox et importez l'image ScadaBR en cliquant sur File ➔ Import Appliance

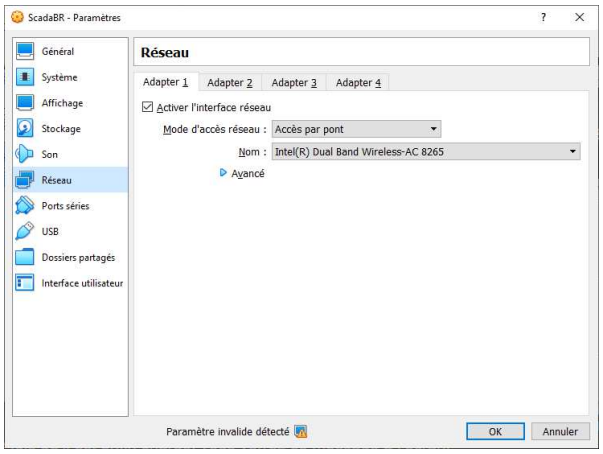


5. Cliquez sur l'icône du dossier et sélectionnez le fichier ScadaBR.ova téléchargé à l'étape 2,



6. Cliquez sur Suivant, puis Importez pour charger l'image ScadaBR dans VirtualBox,
7. Avant de démarrer ScadaBR, vous devez configurer le réseau de votre machine virtuelle afin que ScadaBR puisse voir les automates connectés à votre réseau. Sur la fenêtre principale de VirtualBox, sélectionner la machine ScadaBR et sélectionnez « configuration » 

8. Passez au menu réseau et à l'onglet « Adapter 1 » assurez-vous que **l'accès en pont** est sélectionné. Ensuite, sur « Nom : » choisissez votre adaptateur réseau. VirtualBox va reproduire votre adaptateur réseau pour la machine virtuelle. Par conséquent, assurez-vous que l'adaptateur que vous sélectionnez est en fait l'adaptateur réseau qui est connecté à votre réseau. Une fois que vous avez terminé, cliquez sur OK pour fermer cette fenêtre.

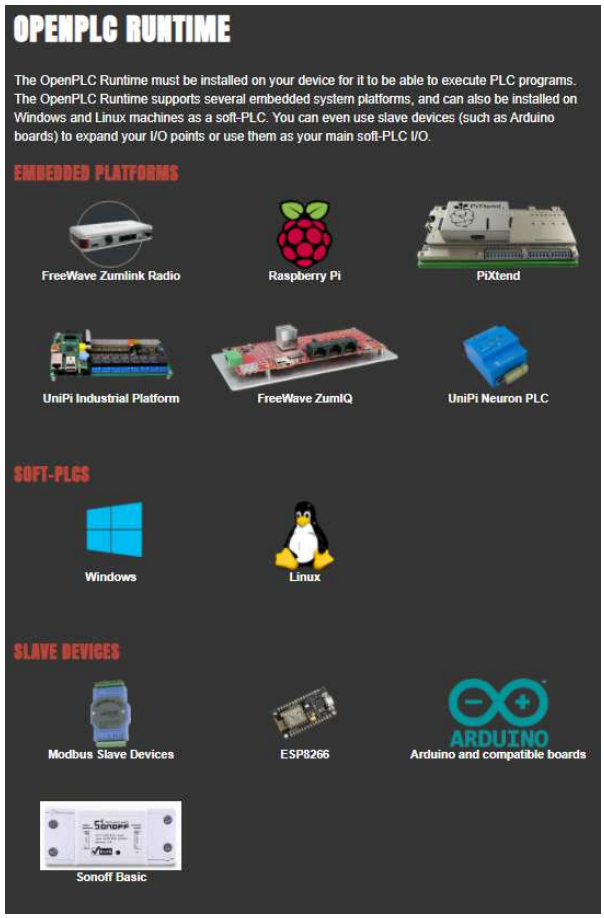


9. Maintenant, vous êtes prêt à y aller ! Sur l'écran principal de VirtualBox, cliquez sur Démarrer pour lancer ScadaBR. Le chargement peut prendre quelques secondes. Une fois qu'il est enfin chargé, une console s'ouvre.

Voilà, j'arrête là pour la partie IHM, ce n'est pas le but de l'article, la documentation fournie est explicite.

III/ Transformer le Raspberry en API

La partie Runtime du projet va permettre d'utiliser un Raspberry comme un automate avec des entrées/sorties TOR.



Mais on peut voir qu'il est possible d'utiliser d'autres architectures, voire même utiliser des solutions logicielles (SOFT-PLCs).
Les SLAVE DEVICES sont à considérer comme des entrées/sorties déportées aux EMBEDDED PLATEFORMS ou SOFT-PLCs. La solution à base d'Arduino sera abordée plus loin dans l'article.

Après avoir cliqué sur Raspberry Pi, on obtient la démarche à suivre. Je ne détaillerais donc pas la procédure dans son ensemble, il suffit de suivre. Je vais seulement parler de l'installation de Raspberry Pi OS sur la carte SD.

Se rendre sur : Raspberry Pi OS – Raspberry Pi

Télécharger Raspberry Pi Imager, l'installer et l'exécuter. Configurer comme ci-dessous par exemple, afin d'avoir une version légère du sytème :

Une fois installé, je conseille d'activer le SSH, pour cela :

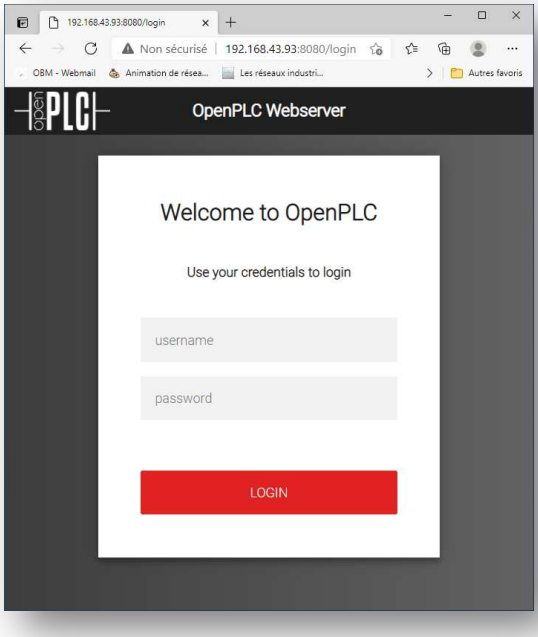
Tant que la carte SD est dans l'ordinateur, sous Windows, ouvrir un éditeur de texte (pas ~~Word~~ c'est un traitement de texte) du type Notepad+ et créer un fichier ***ssh****. Ce fichier ne doit pas avoir d'extension et doit ne rien contenir. Le placer dans le répertoire boot de la SDCARD. Cela, avant la première introduction dans le Raspberry. Ainsi, avec PuTTY et la connaissance de l'IP du Raspberry, il sera possible de s'y connecter depuis votre Windows et disposer du clavier.*

Une fois sous PuTTY et connecté au Raspberry avec nom : pi et pass :raspberry, il suffit de suivre les indications du site, c'est-à-dire :

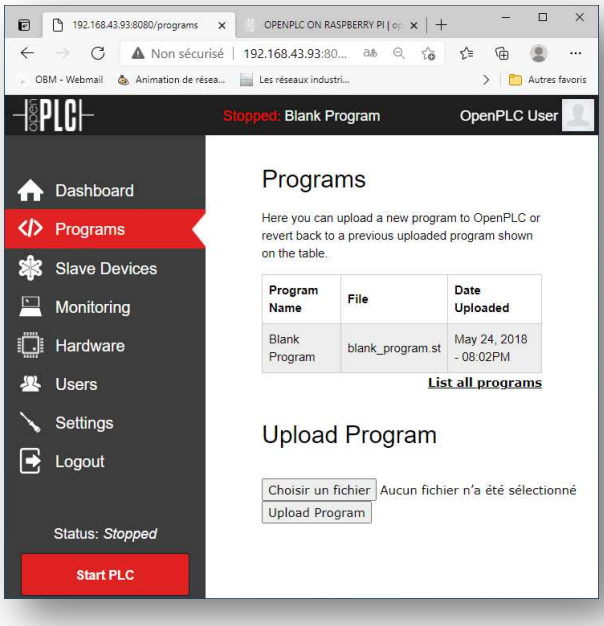
```
git clone https://github.com/thiagoralves/OpenPLC_v3.git
cd OpenPLC_v3
./install.sh rpi
```

Le processus d'installation prendra un certain temps (jusqu'à 1 heure selon votre système). Une fois OpenPLC installé, il suffit de redémarrer l'appareil.

Le Runtime OpenPLC dispose d'un serveur web intégré qui permet de configurer OpenPLC et aussi de télécharger de nouveaux programmes pour qu'il s'exécute. Ce serveur web peut être consulté en ouvrant un navigateur Web sur un ordinateur et en tapant l'adresse IP de l'appareil OpenPLC avec comme port 8080. Par exemple, si votre Raspberry Pi a l'IP 192.168.43.93 sur votre réseau, vous devez taper ceci sur votre navigateur :



Le nom d'utilisateur et le mot de passe par défaut sont **openplc** (login) et **openplc** (mot de passe).

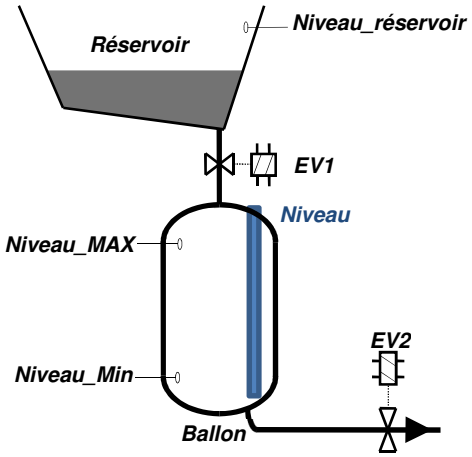


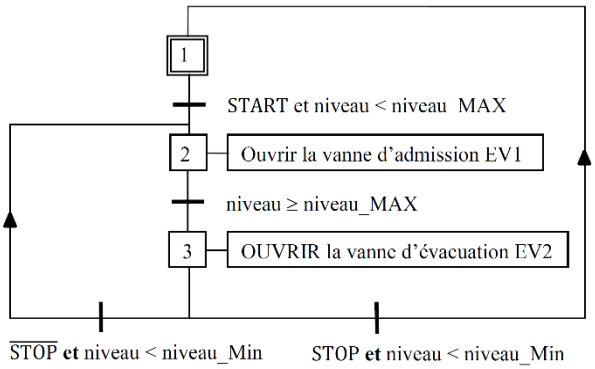
Ça y est, on a l'automate actif, on va pouvoir y implanter notre programme.

IV/ Programme implémenté dans l'API

IV.1/ Cahier des charges

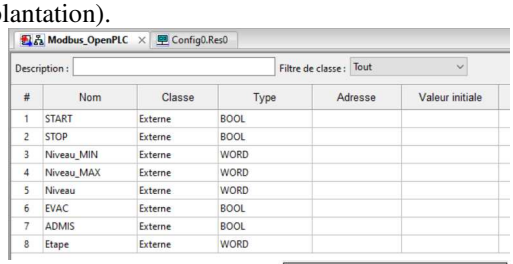
L'application consiste à simuler le contrôle du niveau d'un liquide dans un réservoir :



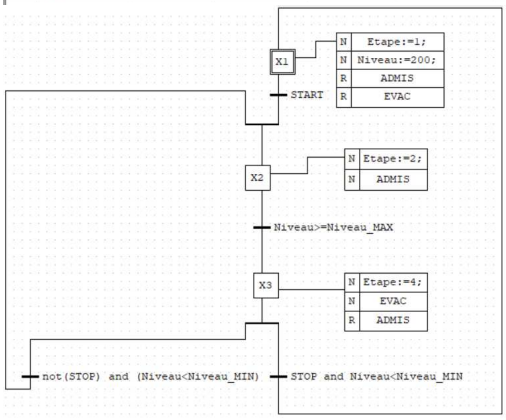


IV.2/ Programme API

La programmation sous OpenPLC de ce Grafcet donne le programme suivant (Se référer à l'article du numéro 103 de la Revue 3EI pour avoir de l'aide sur son implantation).



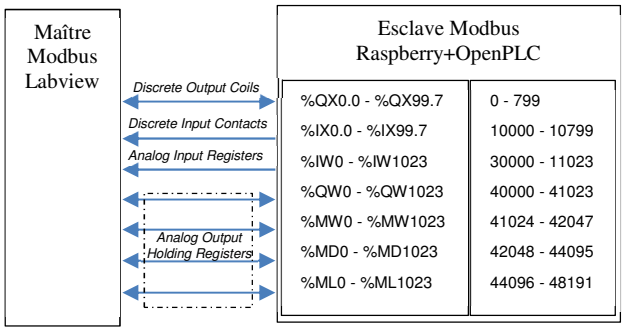
#	Nom	Classe	Type	Adresse	Valeur initiale
1	START	Externe	BOOL		
2	STOP	Externe	BOOL		
3	Niveau_MIN	Externe	WORD		
4	Niveau_MAX	Externe	WORD		
5	Niveau	Externe	WORD		
6	EVAC	Externe	BOOL		
7	ADMIS	Externe	BOOL		
8	Etape	Externe	WORD		



Sachant que les variables ci-dessus seront échangées sur le réseau Modbus/TCP, elles seront déclarées en externe.

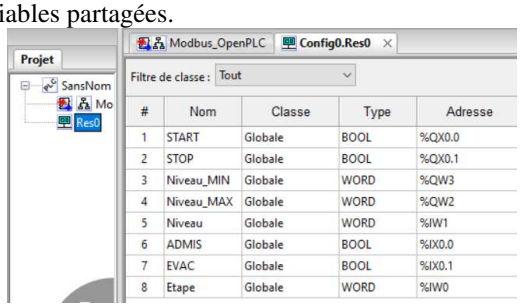
IV.3/ Communication Modbus/TCP

Sous l'API OpenPLC (le Raspberry), l'adressage Modbus suit le tableau suivant donné sur le site du développeur.



Pour rappel, un coil est un élément binaire, et un Registre est en un ensemble de bits, soit un mot.

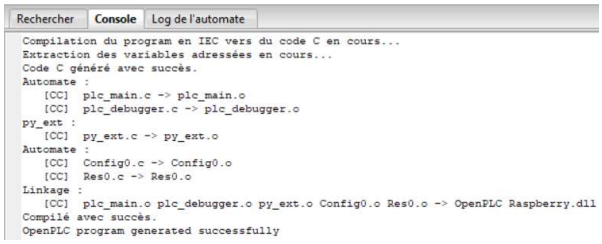
Pour cela, sous l'éditeur OpenPLC, dans l'architecture du projet, **dans Res0**, il faut indiquer l'adressage Modbus. Les variables doivent être de classe Globale pour être visible par l'ensemble des programmes.
Il faut bien faire attention à ce que les types des variables soient identiques entre les variables du programme et les variables partagées.



#	Nom	Classe	Type	Adresse
1	START	Globale	BOOL	%QX0.0
2	STOP	Globale	BOOL	%QX0.1
3	Niveau_MIN	Globale	WORD	%QW3
4	Niveau_MAX	Globale	WORD	%QW2
5	Niveau	Globale	WORD	%IW1
6	ADMIS	Globale	BOOL	%IX0.0
7	EVAC	Globale	BOOL	%IX0.1
8	Etape	Globale	WORD	%IW0

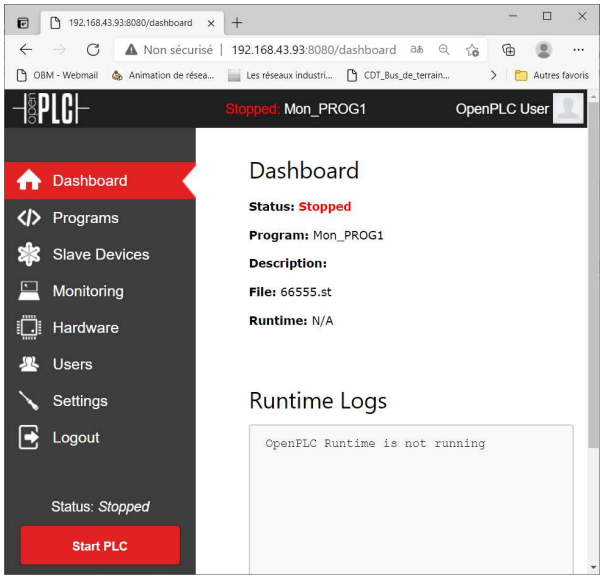
IV.4/ Implanter le programme

Compiler le programme  et valider qu'aucune erreur de programmation n'est présente, comme indiqué ci-dessous :

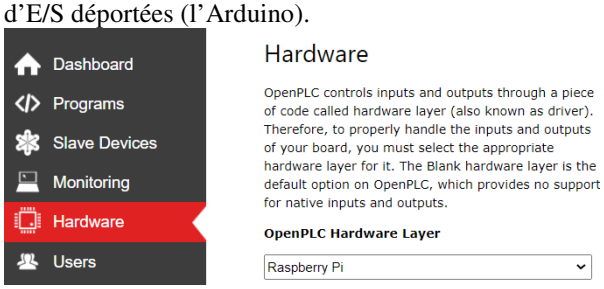


Le programme qu'il faut implémenter dans l'API sera le fichier MAIN.st (nom que j'ai donné).

Ensuite, il faut se connecter sur l'API (Raspberry) à l'aide d'un navigateur Web. Pour cela, il suffit de saisir l'adresse IP de l'automate et le port 8080. Il est évident que l'ordinateur et l'API doivent être sur le même réseau !

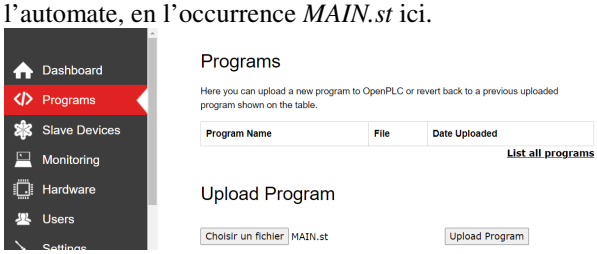


La première chose à réaliser est de définir le matériel (Hardware), ici c'est un Raspberry connecté à une carte d'E/S déportées (l'Arduino).

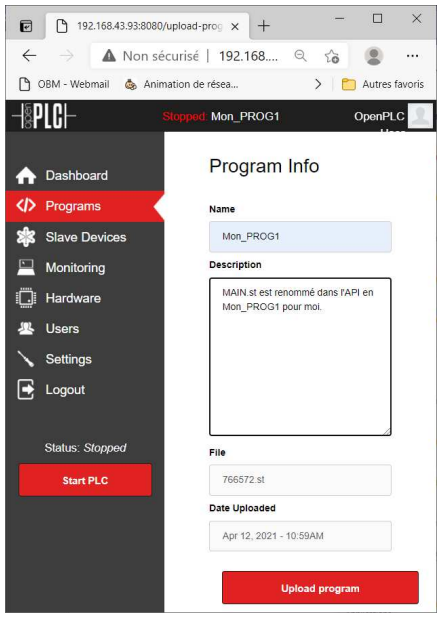


Sauver la configuration définie en cliquant sur *Save Changes* en bas de page.

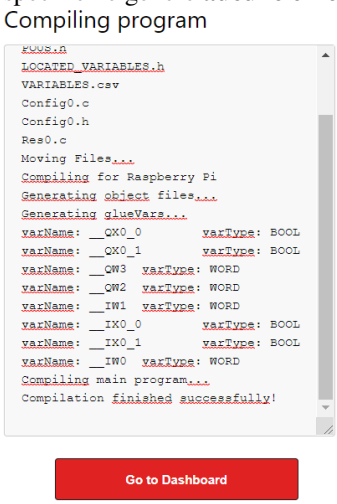
Rentrer dans la section **</>Programs** afin d'ouvrir la page suivante et pouvoir sélectionner le programme de l'automate, en l'occurrence *MAIN.st* ici.



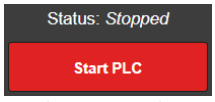
Ensuite, il suffit de cliquer sur *Upload Program* pour transférer le programme dans l'automate. Avant, il est demandé de saisir le nom que l'on souhaite donner au programme dans l'API, dans mon cas Mon_PROG1.



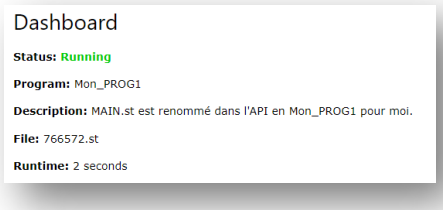
Il faut bien vérifier que le transfert et la traduction vers le hardware spécifié ne génère **aucune erreur**.



Dans le bas du bandeau de gauche, il suffit alors de cliquer sur le bouton Start PLC.



La page web d'exécution apparait.



Il est possible de superviser les variables du programme en cliquant sur le menu *Monitoring* du bandeau gauche.

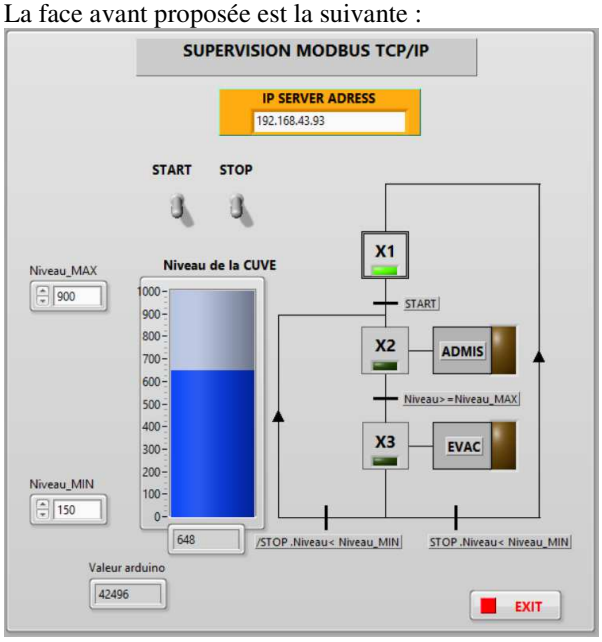
Monitoring

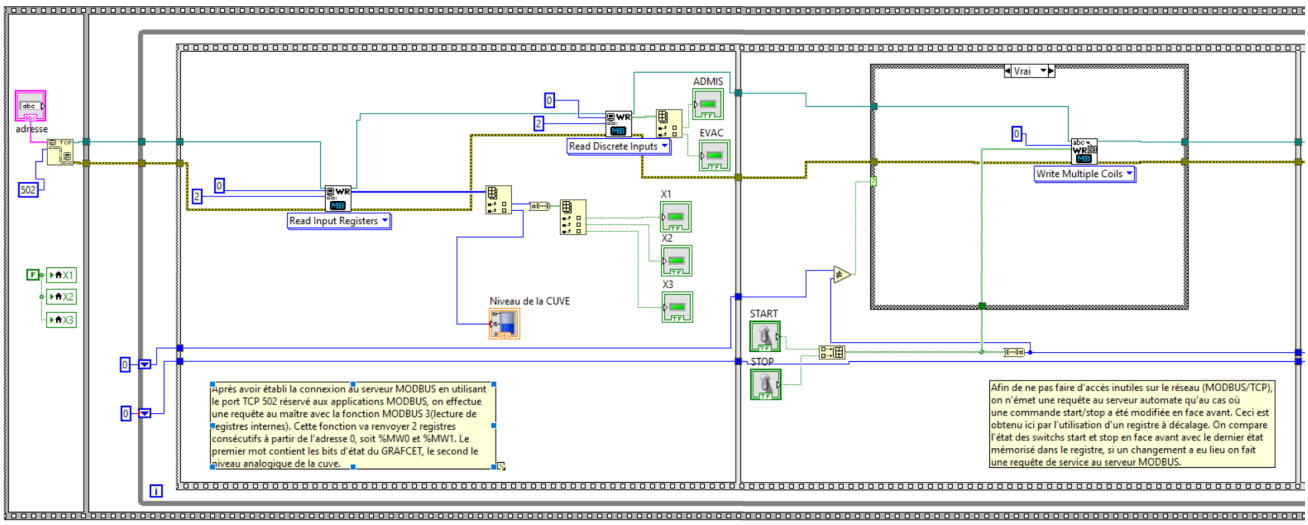
Refresh Rate (ms): 100 Update

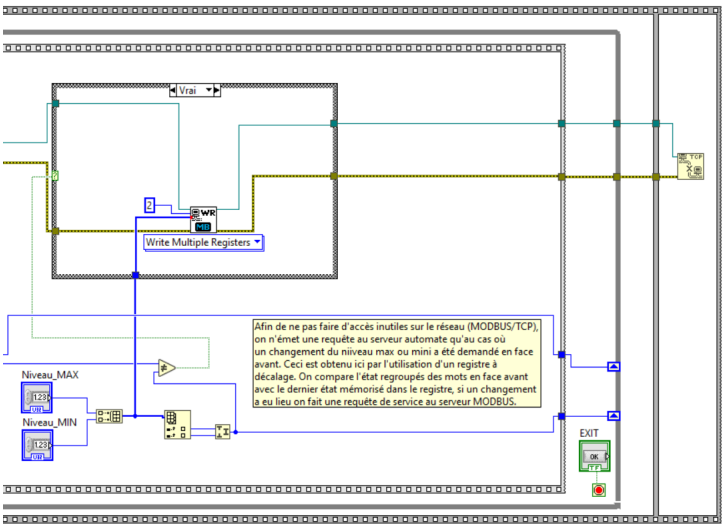
Point Name	Type	Location	Forced	Value
START	BOOL	%QX0.0	No	FALSE
STOP	BOOL	%QX0.1	No	FALSE
Niveau_MIN	WORD	%QW3	No	0
Niveau_MAX	WORD	%QW2	No	0
Niveau	WORD	%IW1	No	200
ADMIS	BOOL	%IX0.0	No	FALSE
EVAC	BOOL	%IX0.1	No	FALSE
Etape	WORD	%IW0	No	1

IV.5/ Programme Labview

Maintenant, la supervision du programme PLC, à travers une interface Labview va être abordée.
La face avant proposée est la suivante :

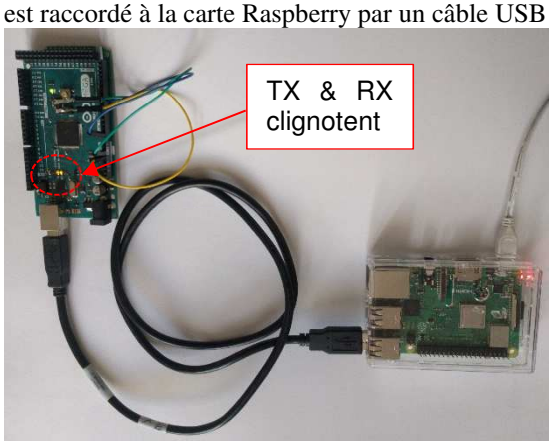






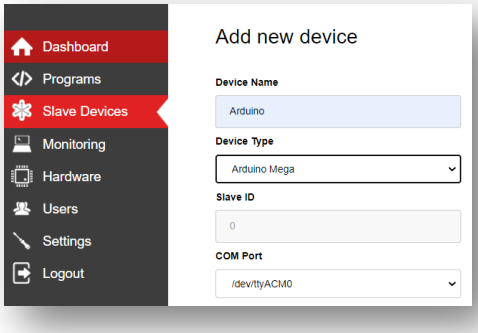
V/ E/S déportées

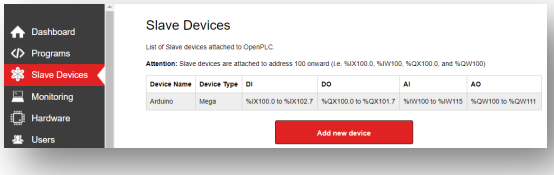
Les entrées et sorties déportées, mais aussi des entrées analogiques, vont pouvoir être réalisées grâce à une carte Arduino. Ici, j'ai utilisé une carte Mega. Celle-ci est raccordé à la carte Raspberry par un câble USB



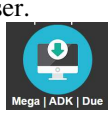
Raspberry+Arduino avec un potentiomètre sur A0

Pour déclarer cette carte Arduino dans l'architecture de l'API, il suffit de se rendre dans le menu « Slave Devices » et d'effectuer la configuration suivante :





Ensuite, récupérer sur le site d'OpenPLC le programme Arduino nécessaire à sa transposition en entrées/sorties déportées et le téléverser.

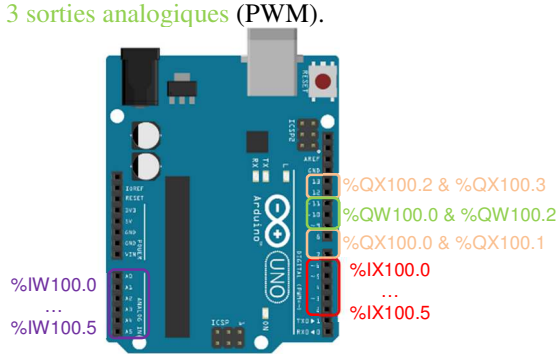


Pour savoir si le port COM est bien déclaré et la connexion active entre le Raspberry et l'Arduino, il faut lancer l'API (Start PLC) et vérifier que les DELs TX et RX de l'Arduino clignotent (après avoir téléverser bien sûr).

Ensuite, dans l'API, l'adressage Modbus va être le suivante (pour OpenPLC v3), elle commence à l'adresse 100.

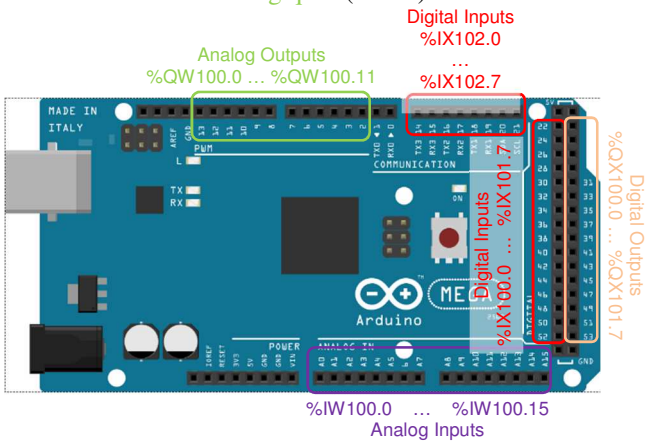
Pour un **Arduino Uno** connecté, on a à disposition :

- 5 entrées TOR,
- 4 sorties TOR,
- 6 entrées analogiques,
- 3 sorties analogiques (PWM).

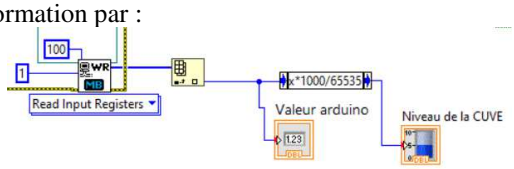


Pour un **Arduino Mega** connecté :

- 24 entrées TOR,
- 16 sorties TOR,
- 16 entrées analogiques,
- 12 sorties analogiques (PWM).



Afin de simuler le capteur de niveau de type analogique, un potentiomètre 10kΩ a été placé sur l'entrée A0, soit l'adresse Modbus %QW100.0.
Sur la supervision LabVIEW, on récupère cette information par :



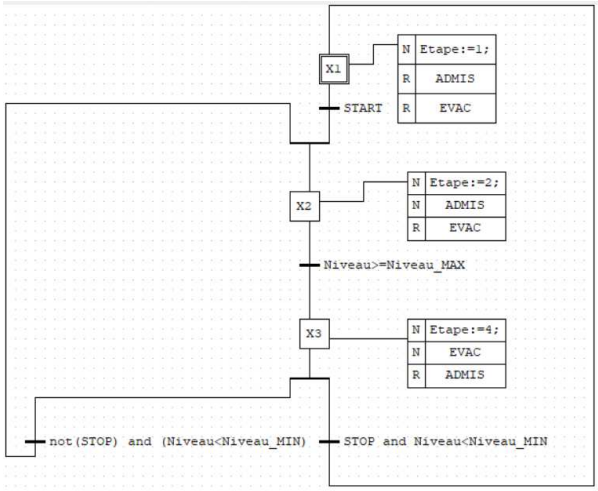
La cuve a un volume de 1000 litres et l'entrée A0 prendra une valeur dans l'ensemble [0,65535]. Cela explique la boite à calcul.

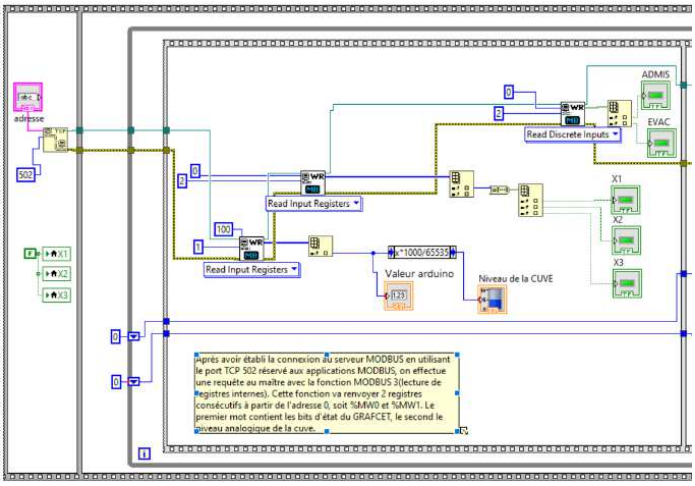
Le programme API est donc à modifier ainsi que le diagramme LabVIEW. Cela donnera les programmes ci-après :

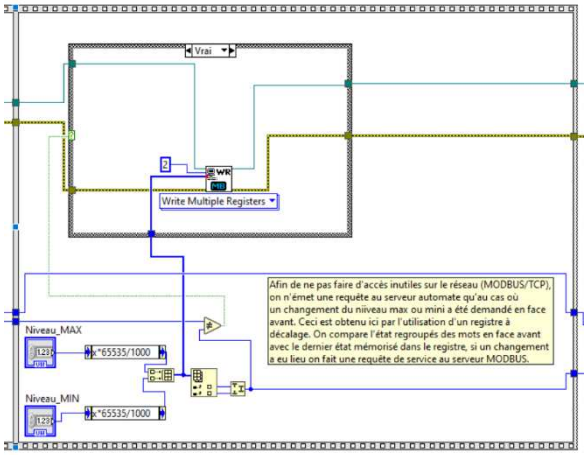
Modbus_OpenPLC | Config0.Res0

Filtre de classe : Tout

#	Nom	Classe	Type	Adresse
1	START	Globale	BOOL	%QX0.0
2	STOP	Globale	BOOL	%QX0.1
3	Niveau_MIN	Globale	WORD	%QW3
4	Niveau_MAX	Globale	WORD	%QW2
5	Niveau	Globale	WORD	%IW100
6	ADMIS	Globale	BOOL	%IX0.0
7	EVAC	Globale	BOOL	%IX0.1
8	Etape	Globale	WORD	%IW0

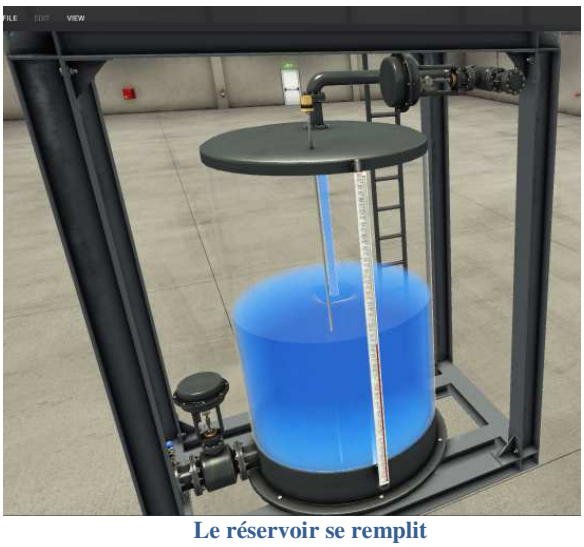






VI/ Pilotage d'un réservoir sous Factory IO

Voici un autre exemple d'application envisageable avec des étudiants. Cette fois-ci, le logiciel Factory IO sera utilisé pour simuler une PO dialguant en Modbus/TCP. Ce logiciel, performant pour simuler une partie opérative plus ou moins complexe permet de rendre « l'expérience » programmation d'API plus visuelle pour l'apprenant, mais aussi plus virtuelle.



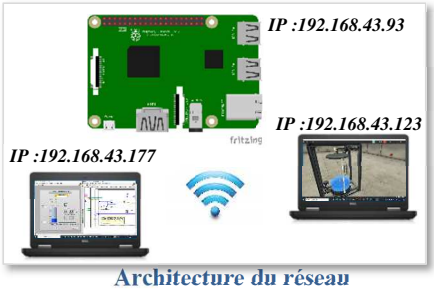
Le réservoir se remplit

Pour cela, il va falloir modifier les programmes vus précédemment. Je ne reprendrai pas tout, ils sont présentés en fin. Je vais juste souligner les modifications nécessaires.

VI.1/ Structure du réseau

L'API (Raspberry) est connecté au réseau par wifi. L'ordinateur n°1 exécutant la supervision sous LabVIEW se trouve sur ce même réseau ainsi que l'ordinateur n°2 qui exécute le logiciel Factory IO. Pour ma démonstration, les deux ordinateurs sont physiquement différents pour davantage de réalisme.

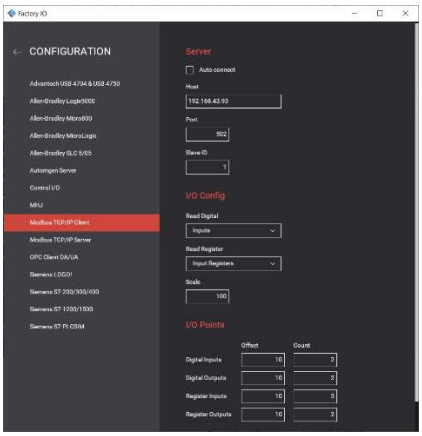
La carte Arduino connectée préalablement n'est plus nécessaire, l'API n'a plus besoin d'E/S déportées. En effet, le logiciel Factory IO simulera le PO avec le module de communication Modbus/TCP connecté à tous les éléments (capteurs et actionneurs).
Pour notre étude, le système est composé d'un réservoir (Tank1), d'une pompe d'admission et une pompe d'évacuation. De plus, un capteur de niveau (0-10V) est présent.



Architecture du réseau

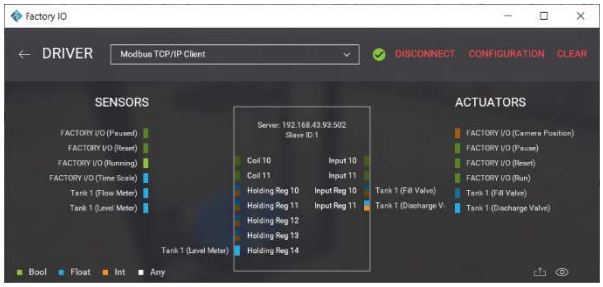
VI.2/ Factory IO

Pour arriver à faire fonctionner le Modbus/TCP, et n'ayant pas un nombre très important de données (8 données) et d'éléments connectés (3 ici), j'ai pris le parti d'appliquer un offset dans l'adresse Modbus/TCP de toutes les données provenant de la PO (Factory IO). Cela donne la configuration suivante pour un driver Modbus TCP/IP Client :



Appuyer sur F4 puis configuration

On remarquera que le logiciel permet d'appliquer un facteur d'échelle par défaut de 100, je l'ai laissé ainsi, ce qui aura une répercussion sur mes programmes LabVIEW et SFC.
Par contre, n'ayant qu'un capteur (le niveau : Level Meter) et deux sorties (les deux pompes) j'ai réduit le nombre d'entrées et sorties comme indiqué ci-dessus.
Je conseille de cocher l'option *Auto connect*.



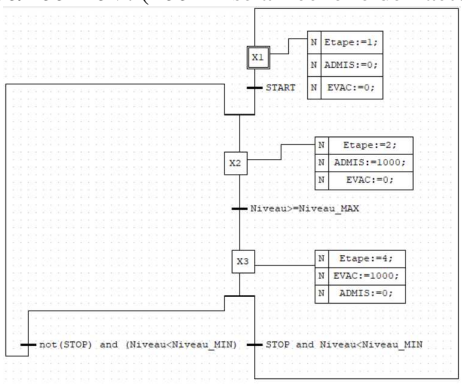
Maintenant, il faut « raccorder » le capteur (virtuel) et les deux actionneurs (les deux pompes) au module Modbus représenté par le rectangle central. Pour cela, il suffit de cliquer sur un sensor et de venir le déposer sur l'entrée souhaitée (ici Holding Reg 14 par exemple). Ce choix aura un impact dans les programmes suivants.

Il suffira de revenir à la vue du système et de faire play ▶ Pour lancer le fonctionnement de la PO.

Remarque : toujours vérifier en bas à droite que vous êtes bien connecté à l'API. Modbus TCP/IP Client

VI.3/ Programme API

La commande des pompes n'est plus binaire (Marche/arrêt) mais proportionnelle (défini dans Factory IO). Ainsi, pour ADMIS=0, la pompe est arrêtée et pour ADMIS=1000, on aura la commande max, soit 1000/100=10V. (100=mise à l'échelle de Factory)



Sinon, l'adresse suivant tient compte de l'adressage défini dans Factory IO.

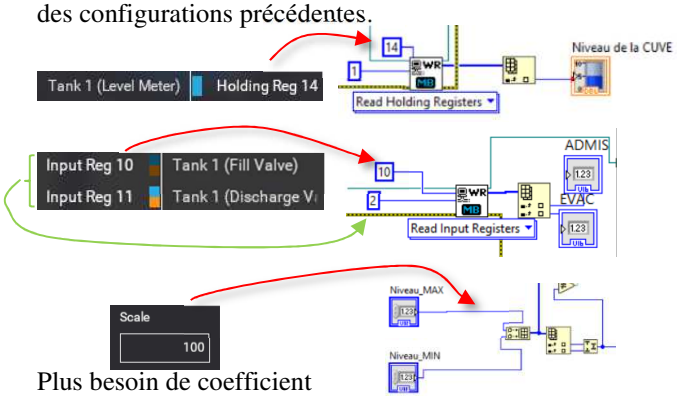
Modbus_OpenPLC | Config0.Res0

Filtre de classe : Tout

#	Nom	Classe	Type	Adresse
1	START	Globale	BOOL	%QX0.0
2	STOP	Globale	BOOL	%QX0.1
3	Niveau_MIN	Globale	WORD	%QW3
4	Niveau_MAX	Globale	WORD	%QW2
5	Niveau	Globale	WORD	%QW14
6	ADMIS	Globale	WORD	%IW10
7	EVAC	Globale	WORD	%IW11
8	Etape	Globale	WORD	%IW2

VI.4/ LabVIEW

Le diagramme LabVIEW n'est que l'aboutissement des configurations précédentes.



Plus besoin de coefficient

VII/ Conclusion

J'espère avoir montré qu'il est possible de faire des activités pédagogiques dans le domaine de l'automatisme avec un Raspberry (et un Arduino). Utiliser ce matériel pour cette activité d'enseignement est, à mon sens, tout aussi industriel que l'utilisation d'un API de type M340 pour beaucoup d'activités.
Vous pourrez retrouver sur Eduscol les programmes LabVIEW et le programme API.
Prenez soin de vous !