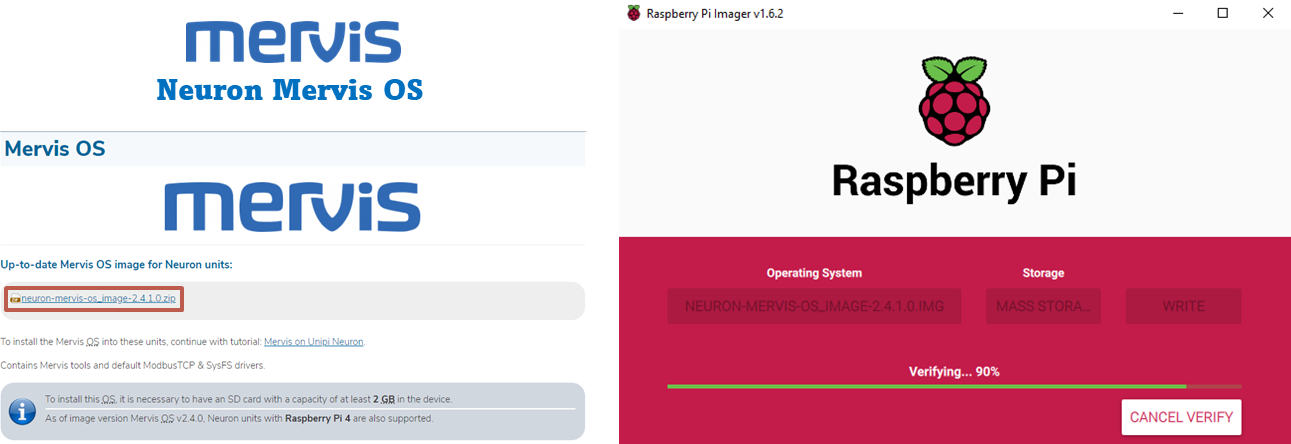
**Les étapes pour transformer un Raspberry Pi avec une extension Unipi 1.1 en PLC :**

* **Installation de l’OS sur le Raspberry Pi**

La version qu’on a installé : <https://kb.unipi.technology/_media/files:software:os-images:unipi1-mervis-os_image-2.4.1.8.zip>

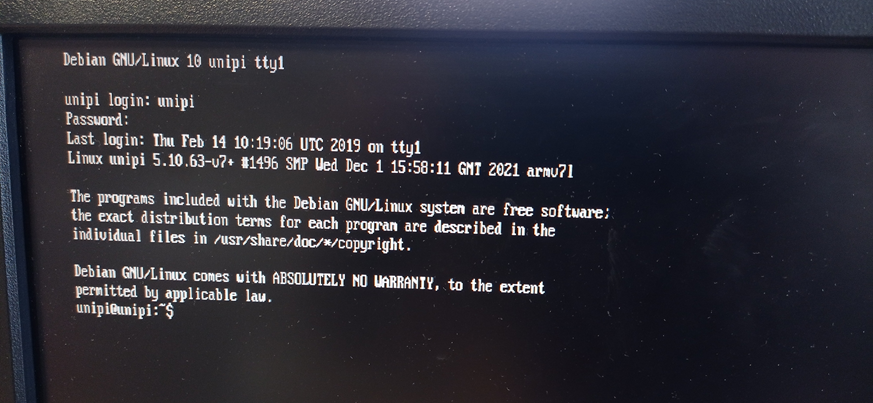
Le clonage de l’OS est fait par : Raspberry Pi Imager



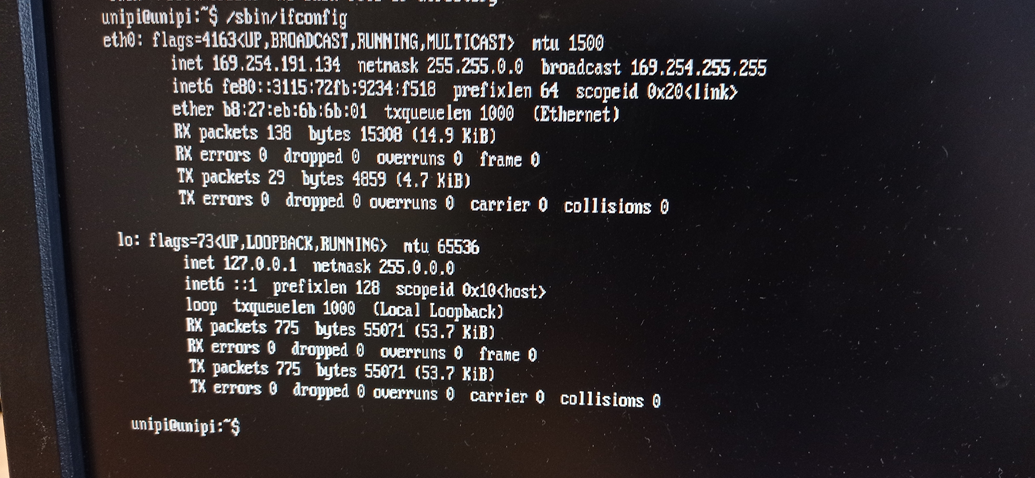
* **Allumons le Raspberry Pi, on peut s’authentifier en utilisant :**

- Login : unipi

- Password : unipi.technology

****

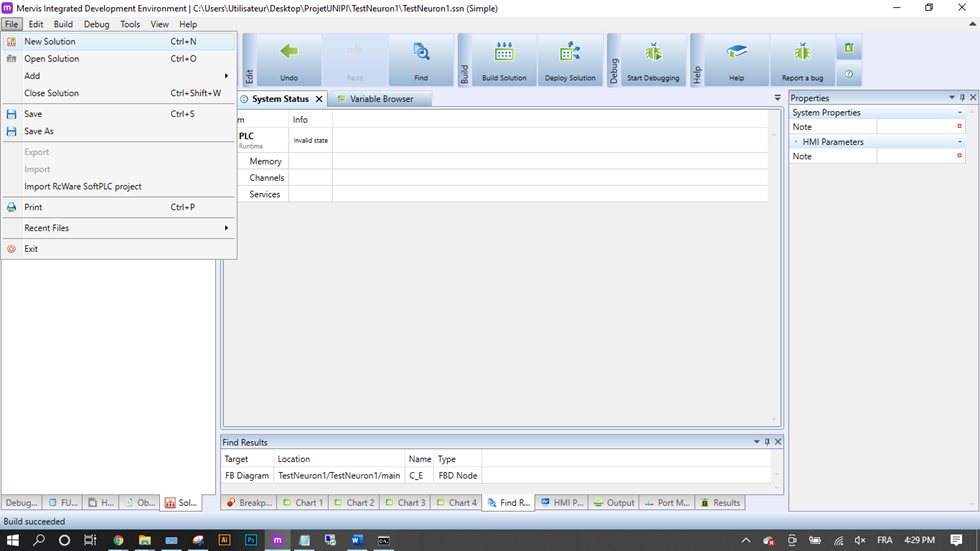
* **On fait une liaison entre le Raspberry Pi et l’ordinateur avec le câble Ethernet, et on récupère l’adresse IP.**

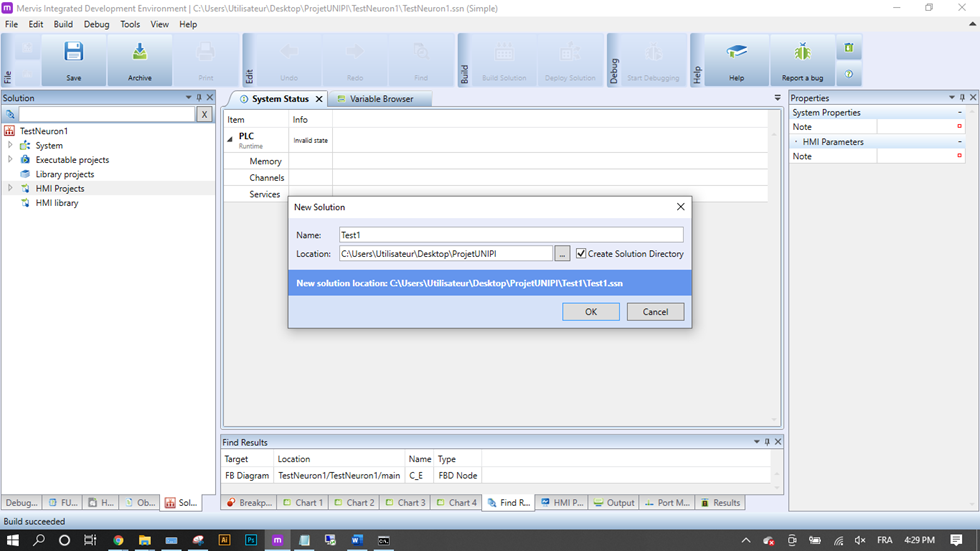


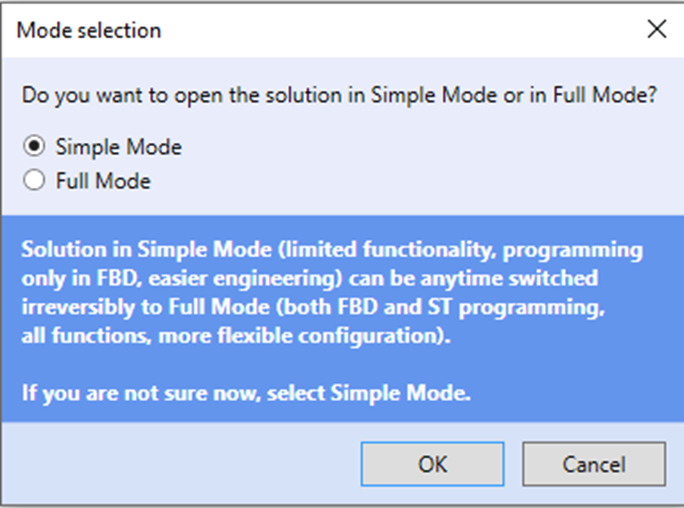
* **Au niveau de l'ordinateur, on va installer Mervis IDE.**



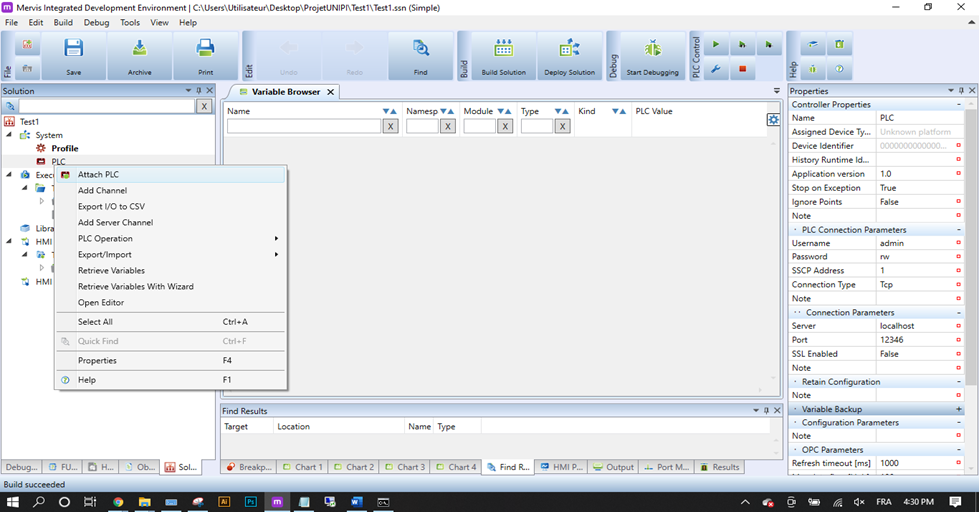
* **On commence par créer une nouvelle solution**

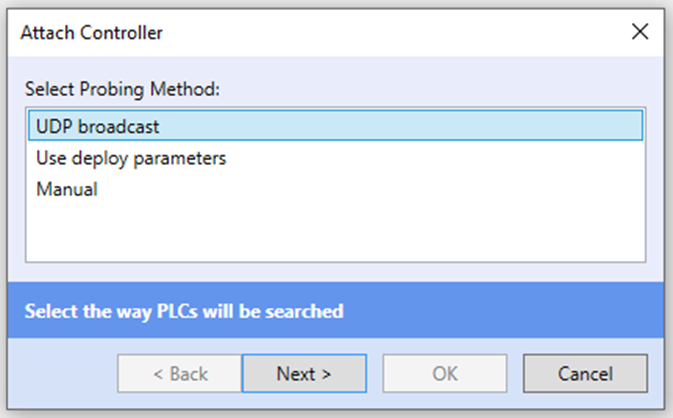


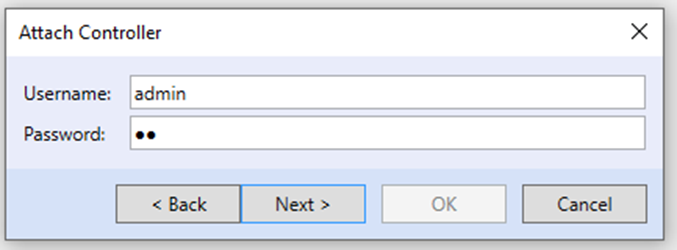


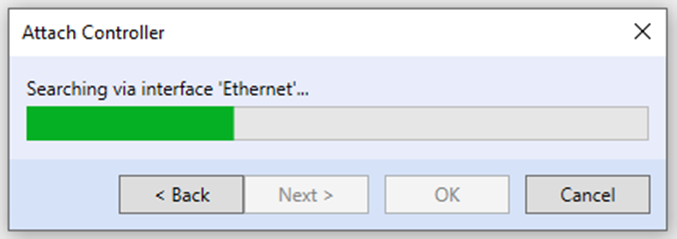


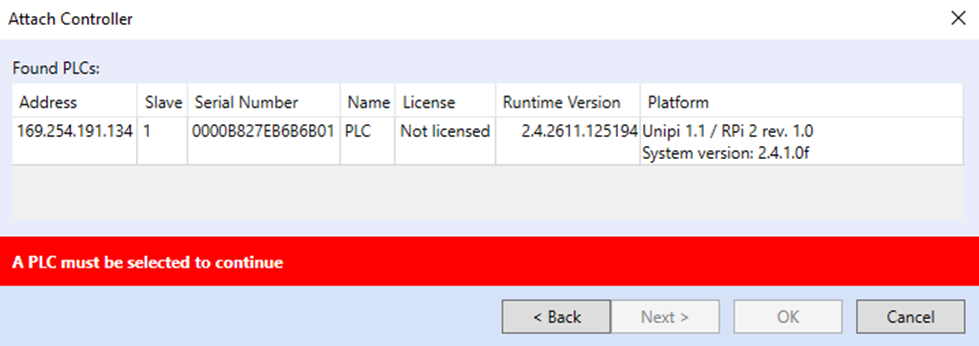
* **Après la création, il faudra que l’IDE détecte le Raspberry Pi qui joue le rôle d’un PLC.**



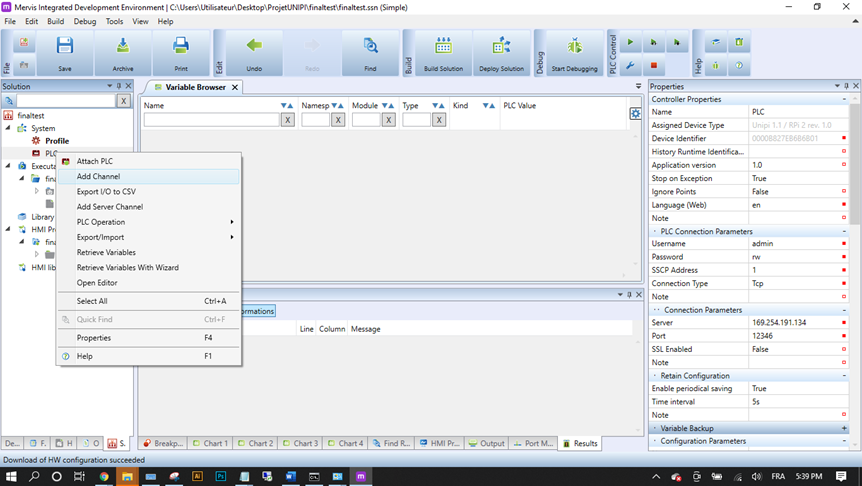


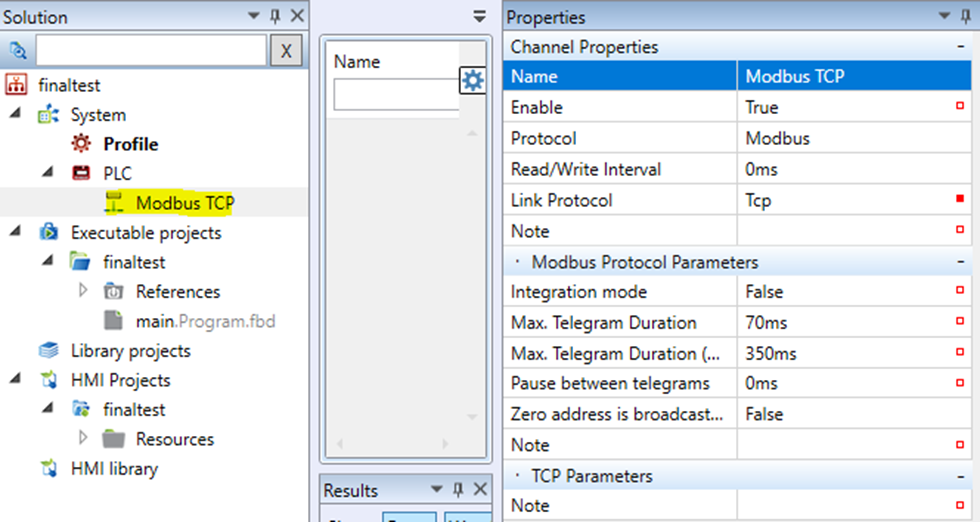




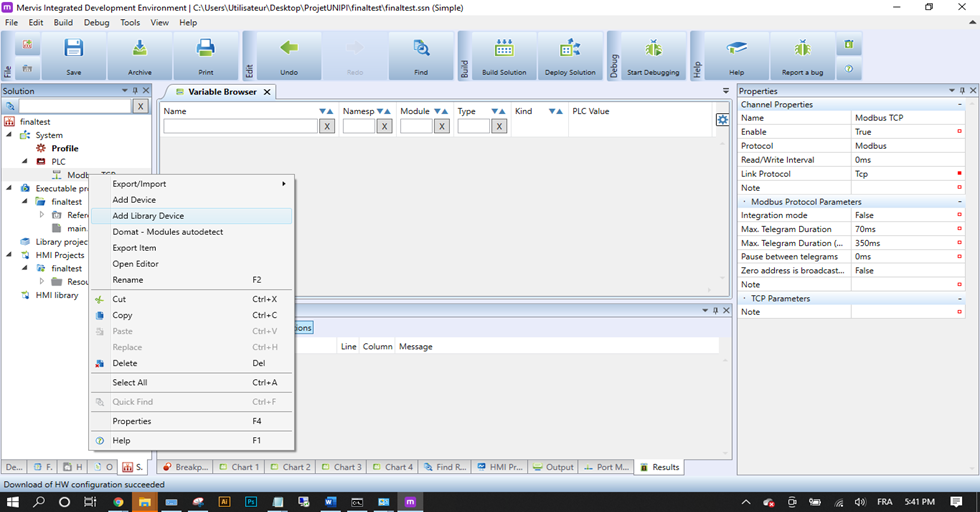


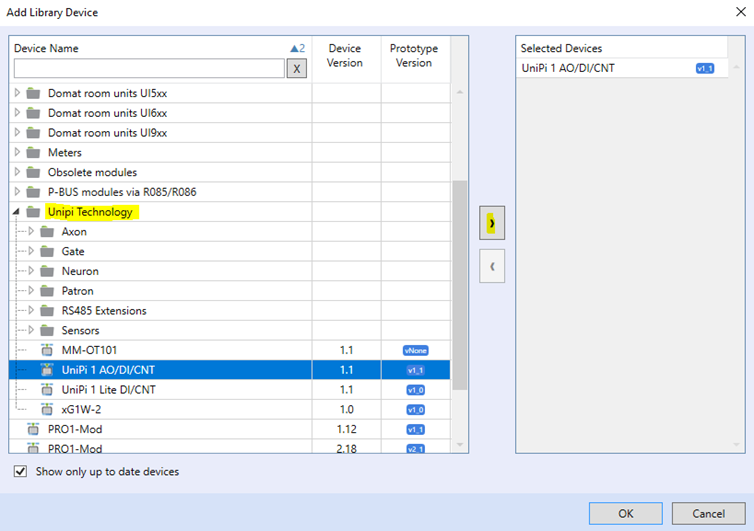
* **Ajoutons un chaine pour la configuration du Modbus TCP**



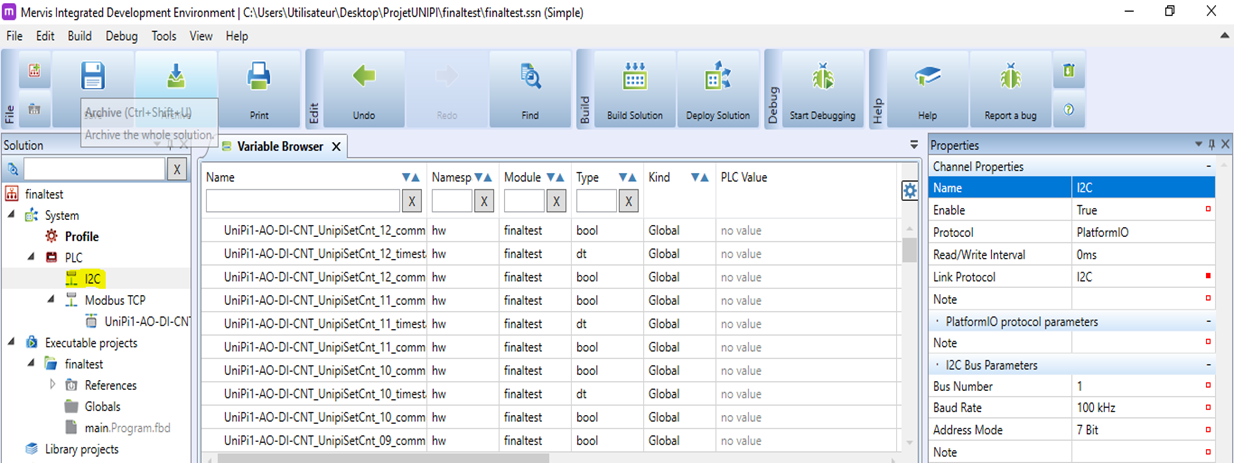


* **Au niveau du chaine de Modbus TCP, on ajoute la bibliothèque des composants correspondants de l’UniPi.**

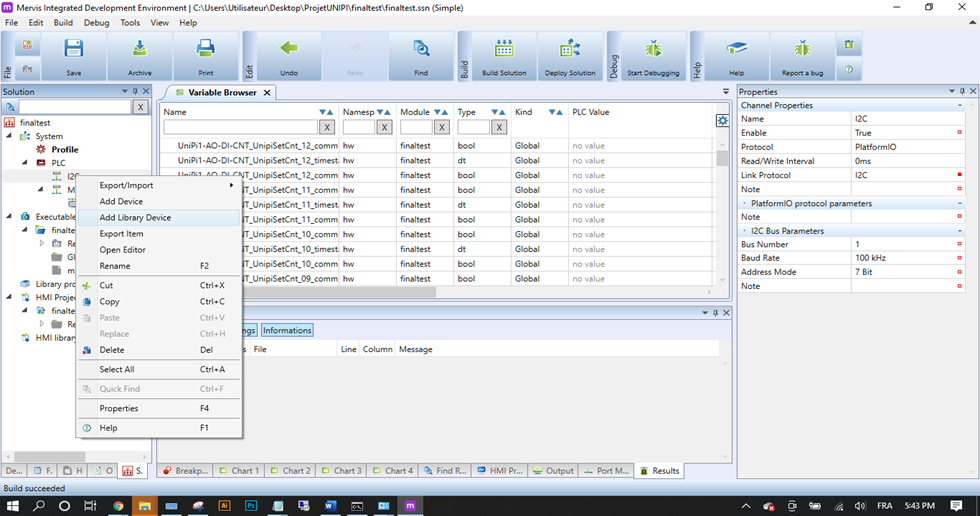


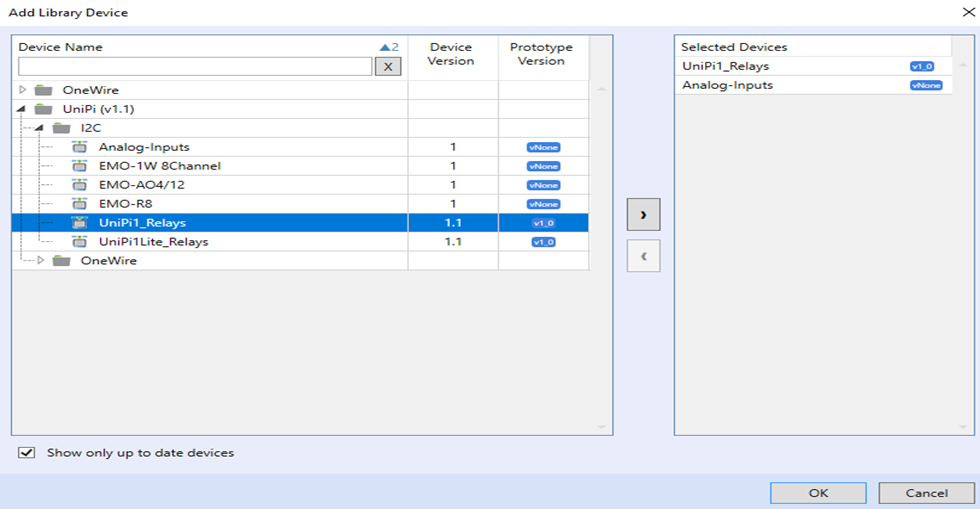


* **On crée une autre chaine I2C.**

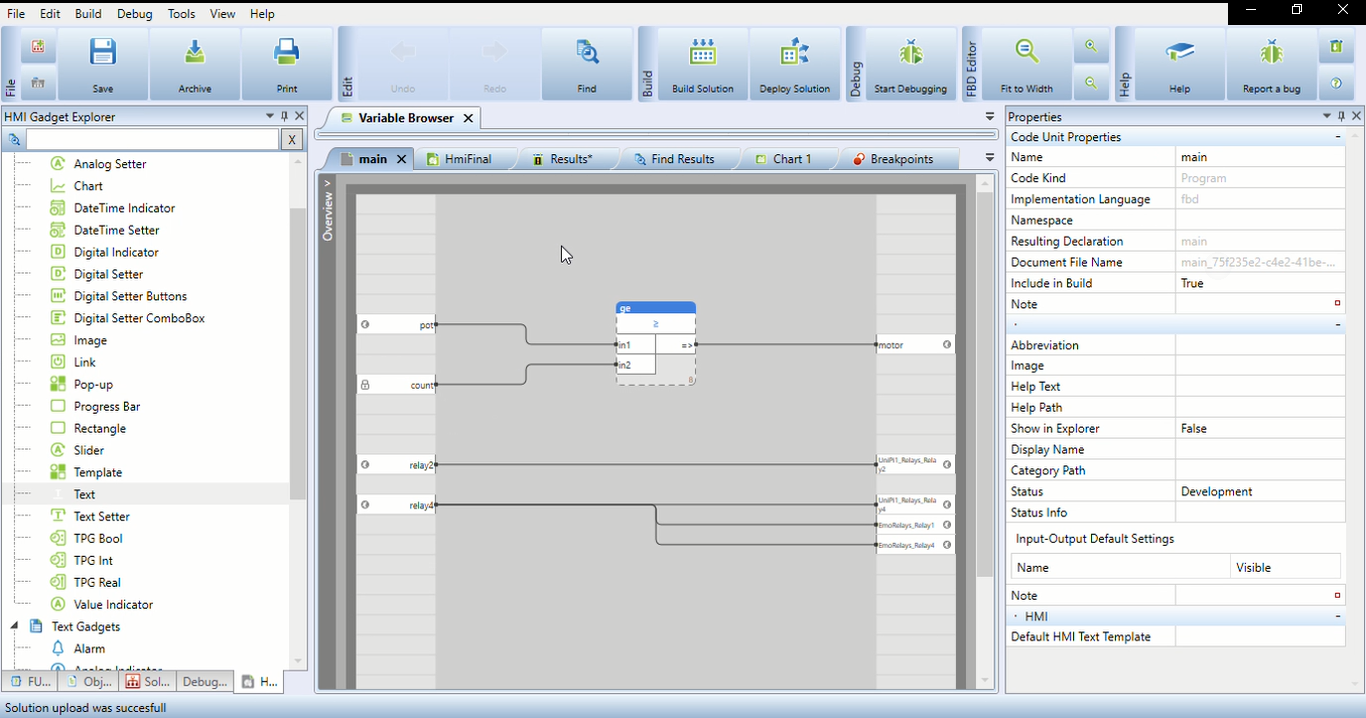


* **Au niveau du chaine de I2C, on ajoute la bibliothèque des composants correspondants de l’UniPi.**

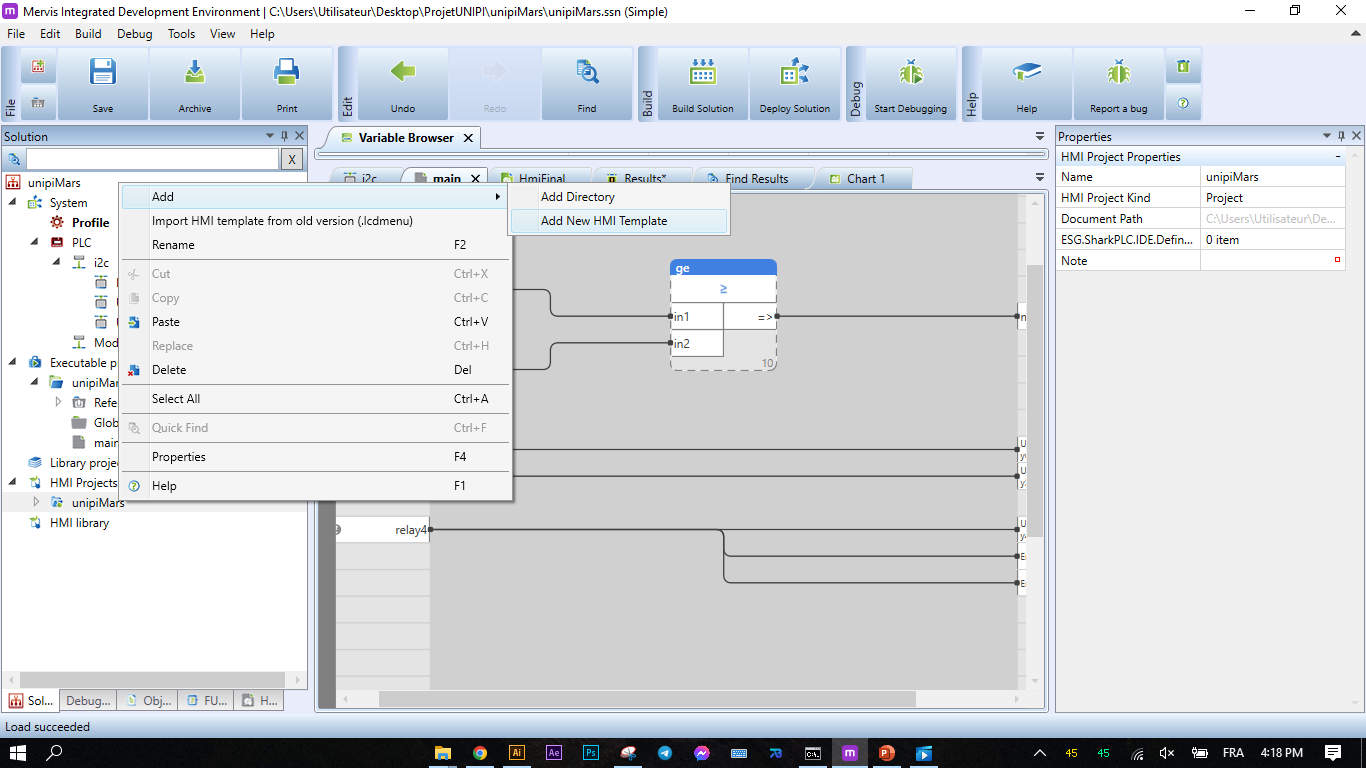


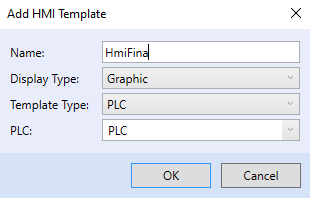


* **On bascule vers le “main” pour créer notre premier programme.**

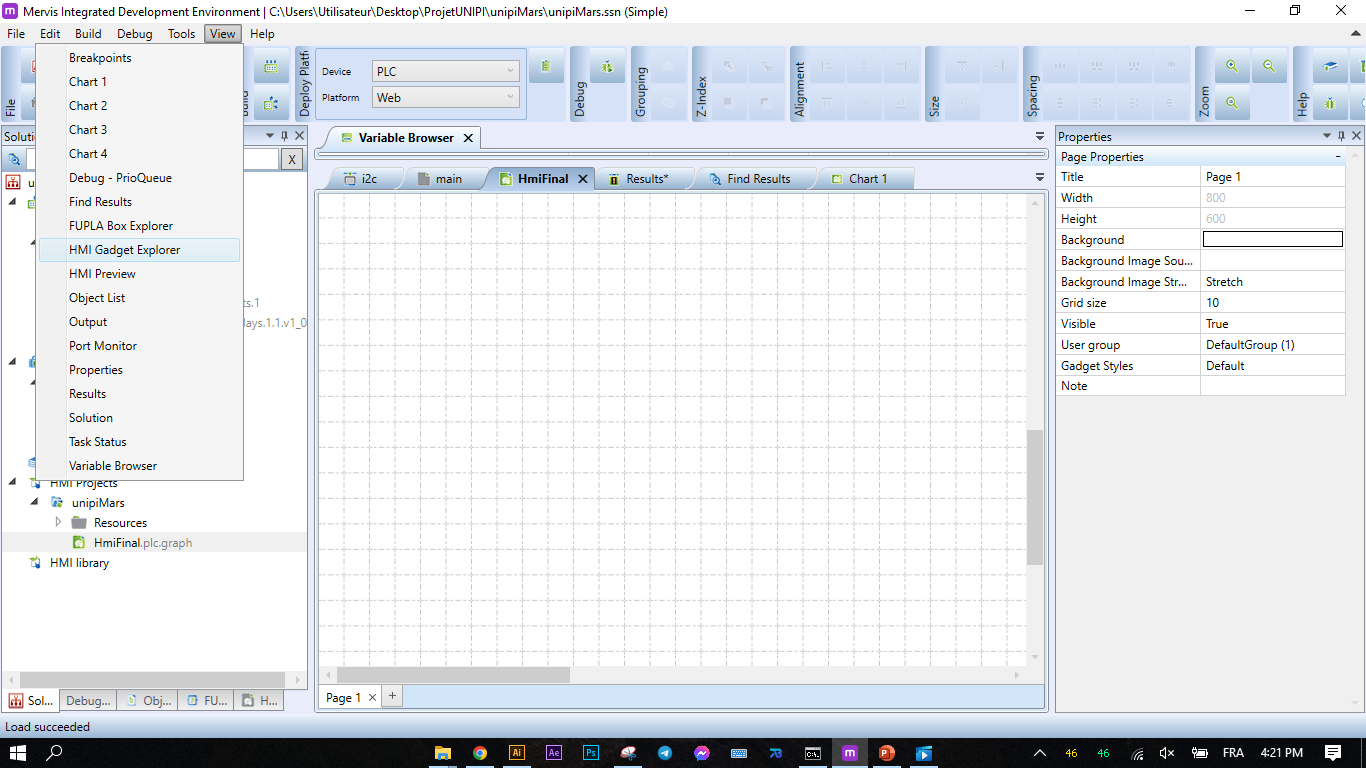


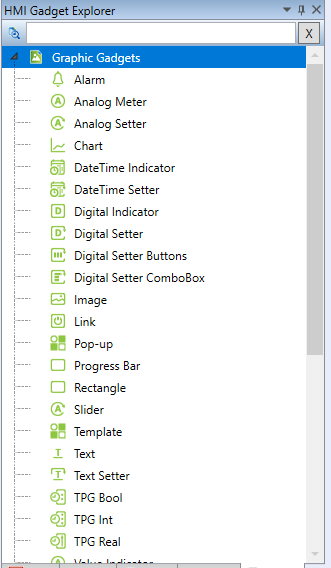
* **On peut créer notre interface homme machine.**



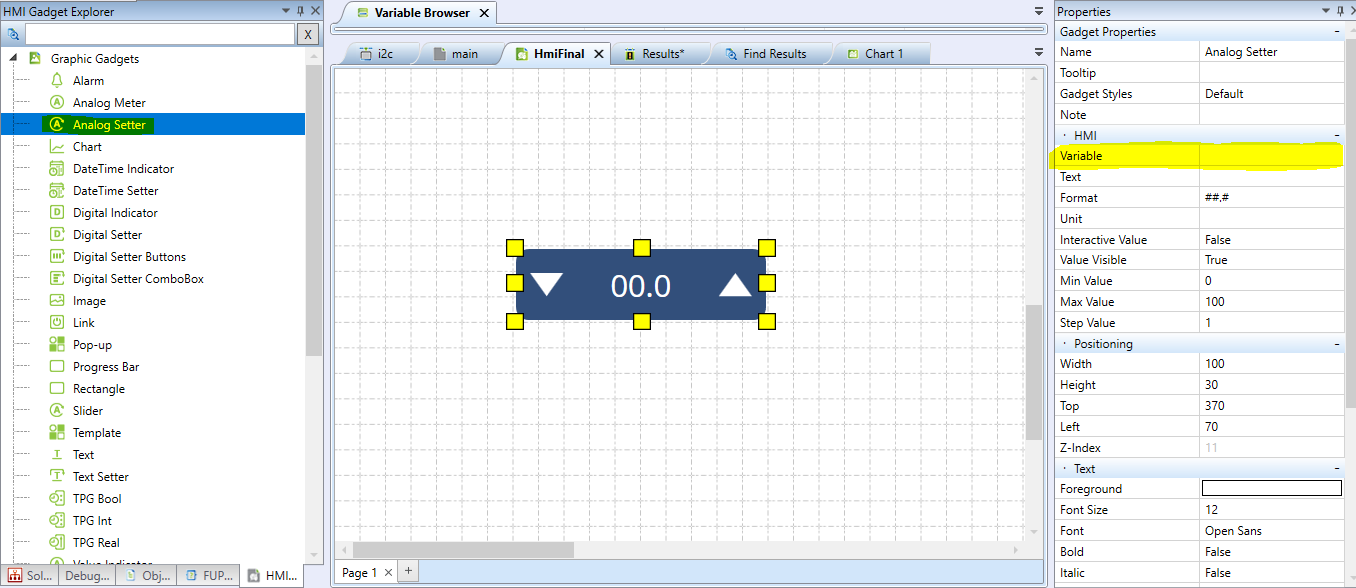


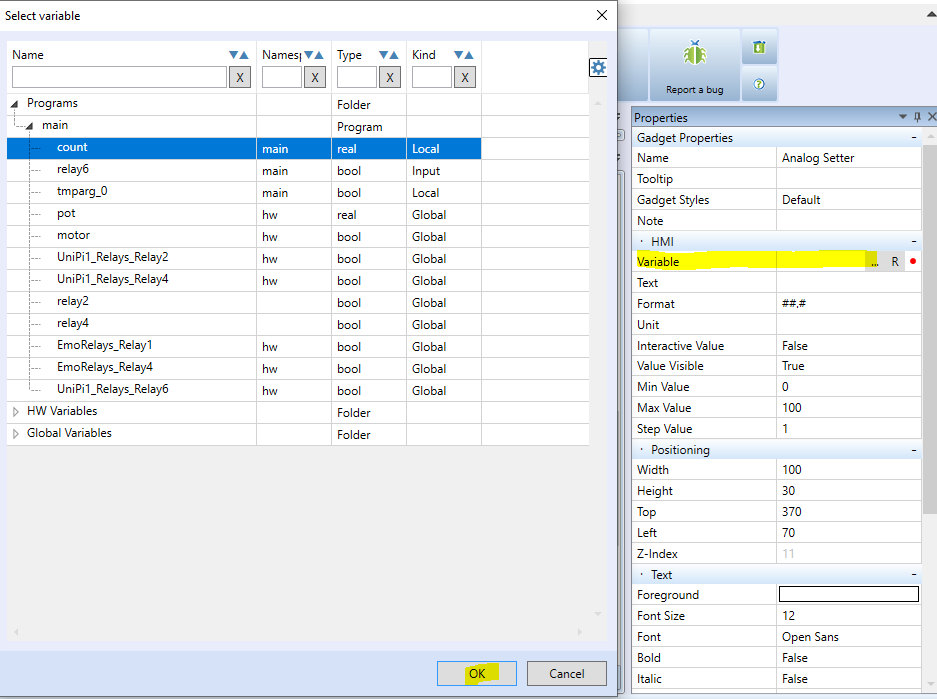
* **On ouvre un menu qui contient des différents éléments qu’on peut utiliser pour la création de notre IHM.**



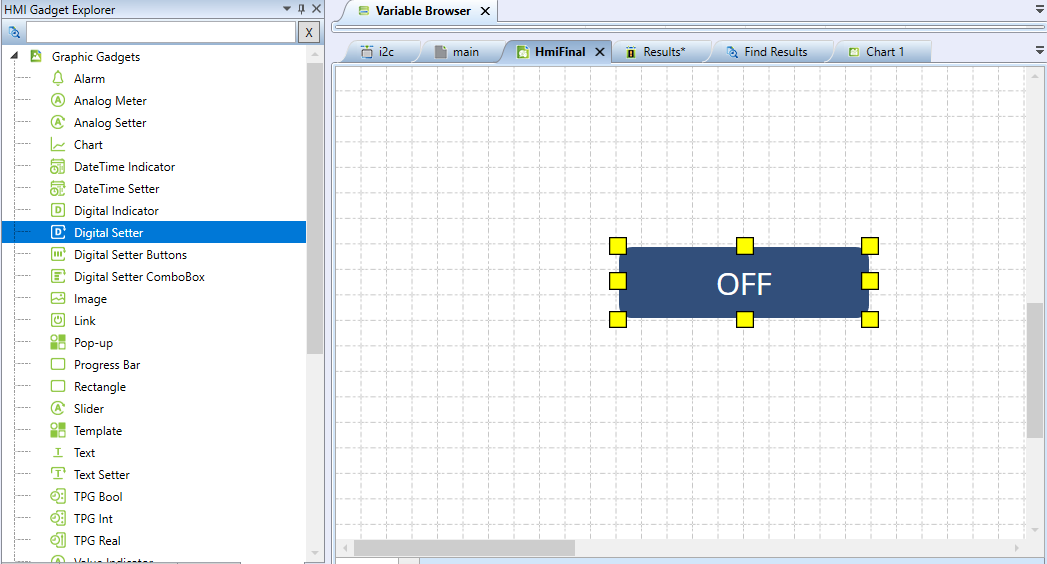


* **On peut placer les élements qui nous intéressent, et les lier automatiquement à une variable.**

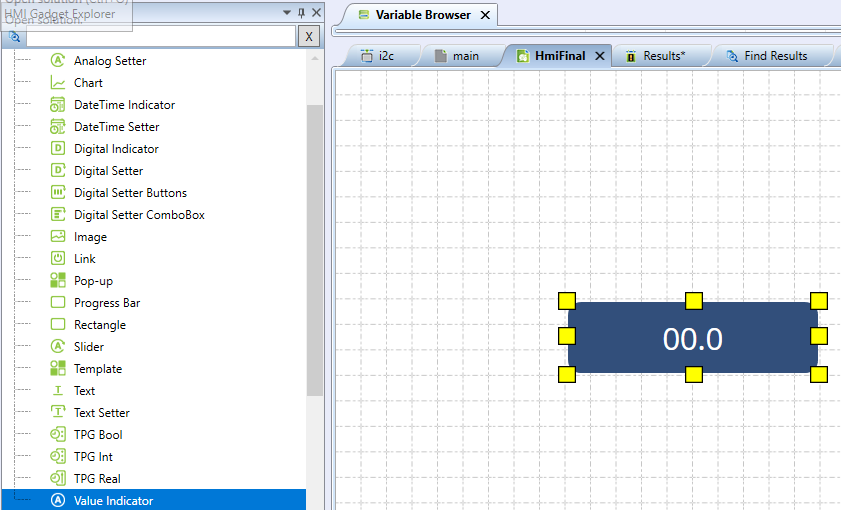
****

****

* **Pour les boutons :**

****

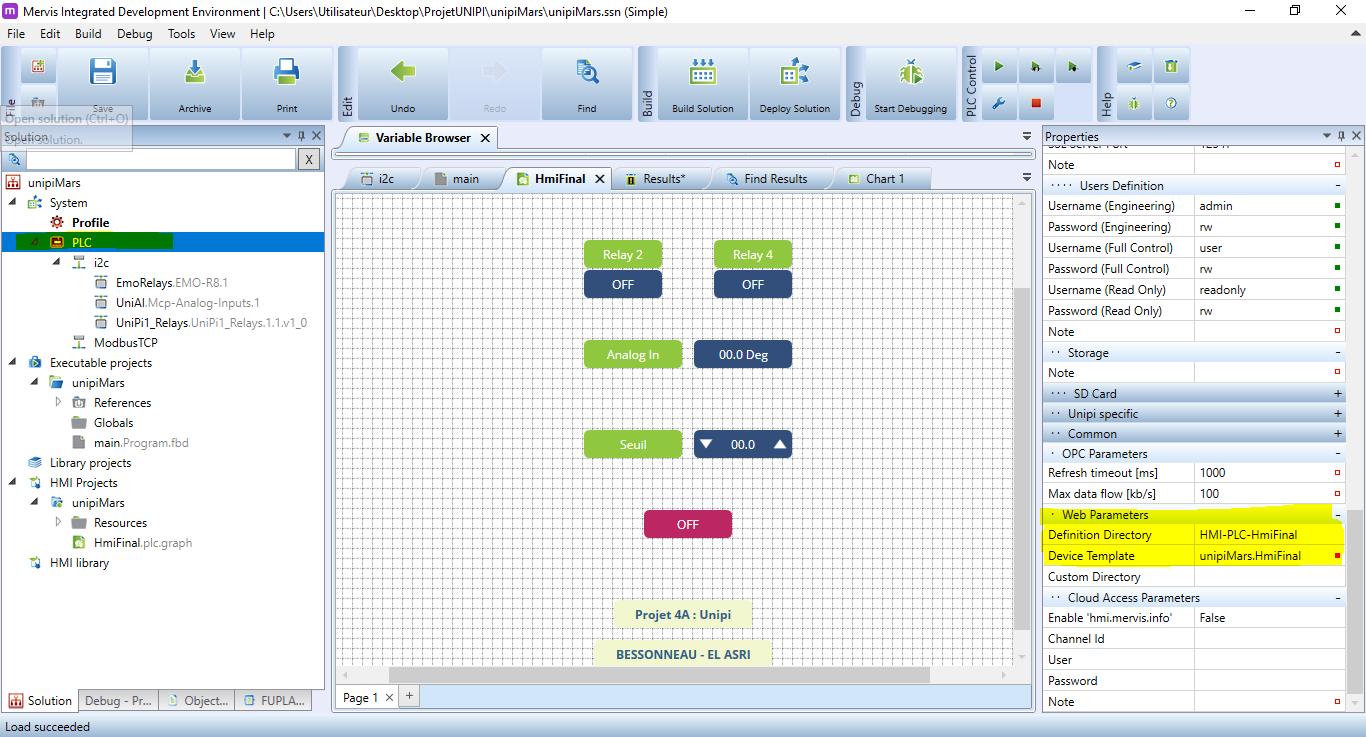
* **Pour lire des valeurs analogiques :**

****

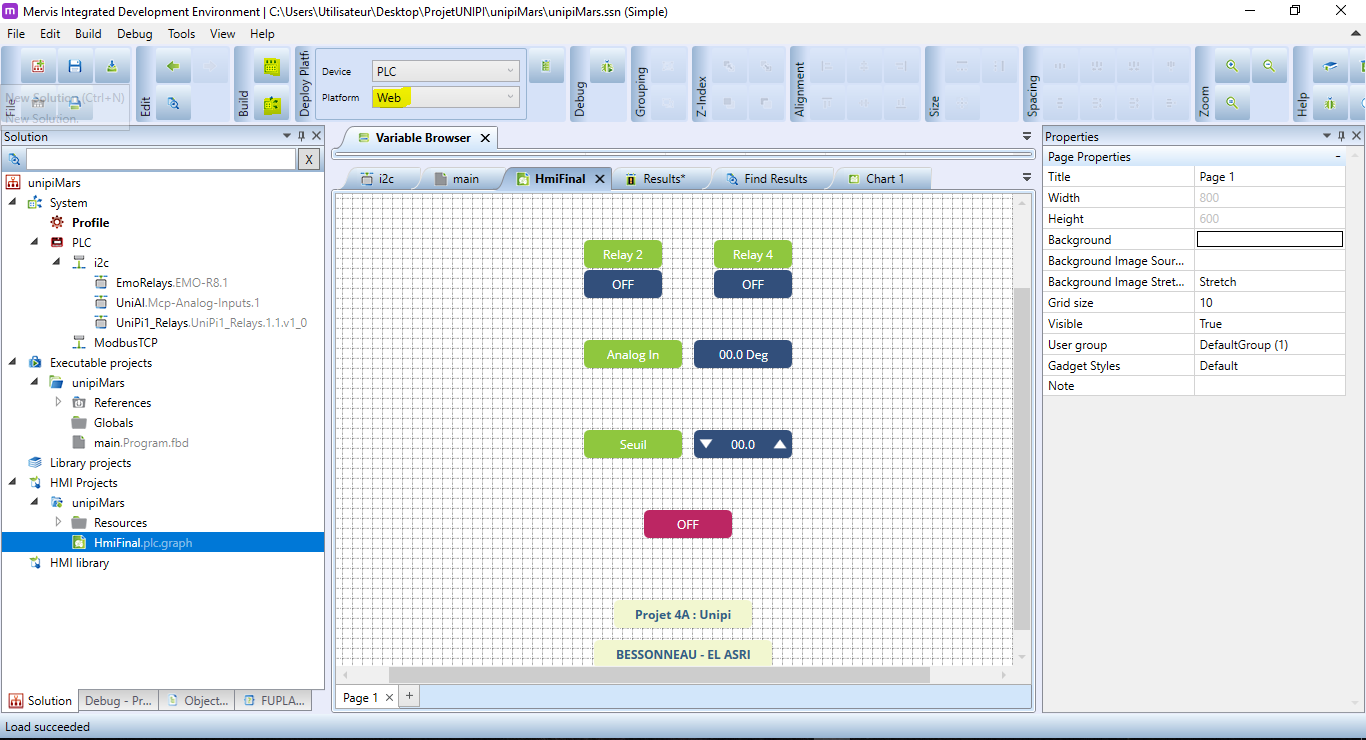
* **Après avoir configuré tous les éléments avec leurs variables appropriées, on obtient notre IHM :**

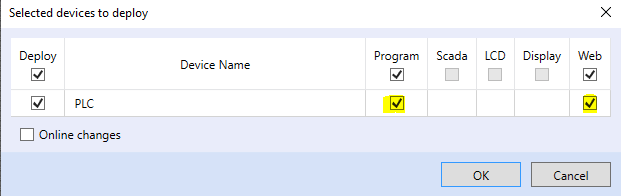
****

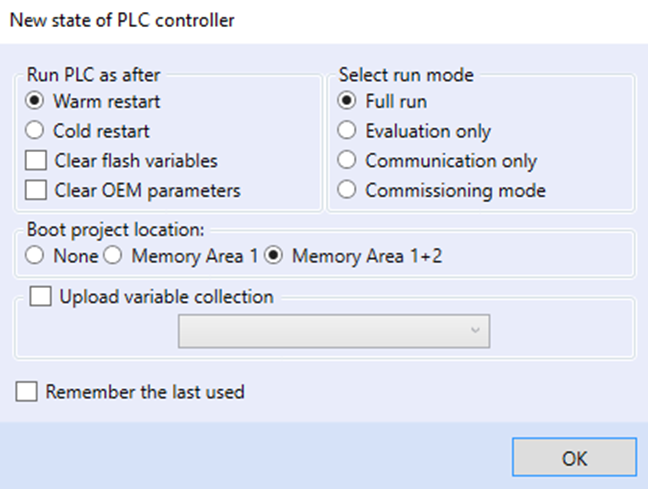
* **Il faut bien vérifié que notre IHM est bien mentionnée au configuration logicielle du PLC :**

****

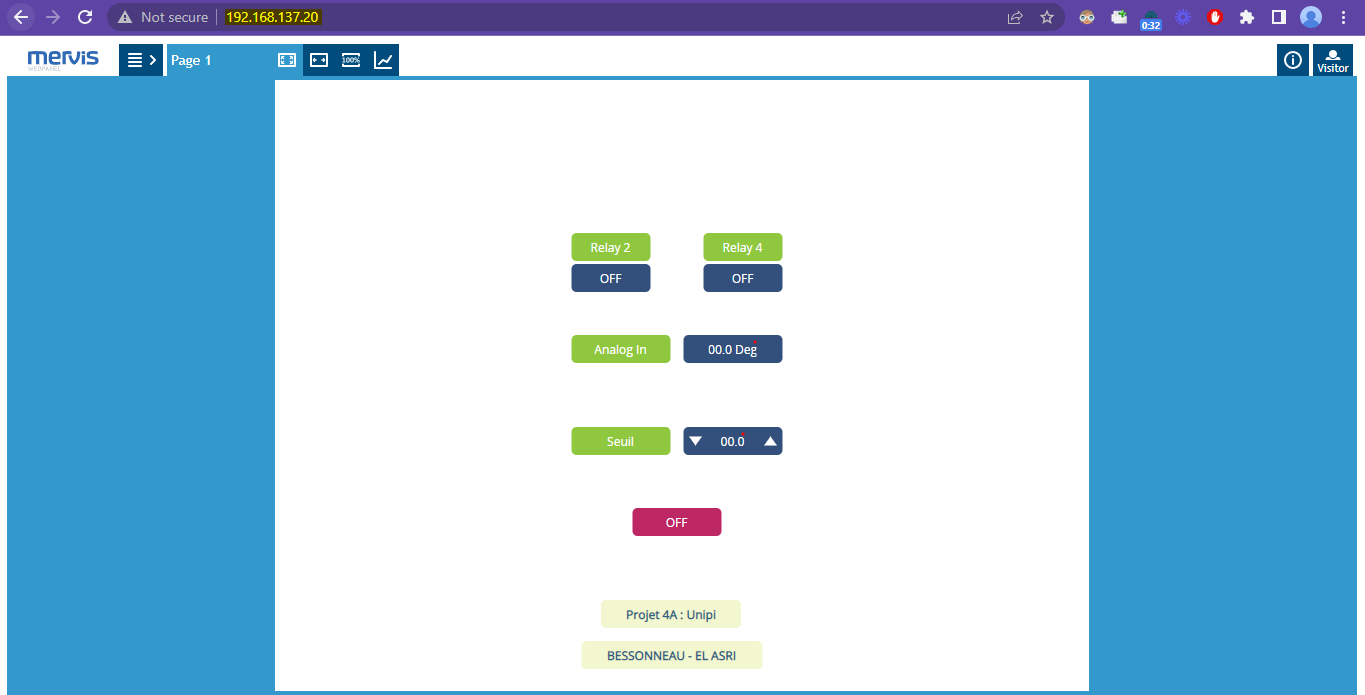
* **On fait le build de notre programme, notre ihm, puis on effectue le déploiement.**

****

****



* **Le programme est bien transmis au Raspberry Pi, et donc notre montage fonctionne, on peut directement visualiser l’IHM grâce à l’adresse IP de la Raspberry Pi :**

****

* **Au niveau de la Raspberry Pi, la première chose qu’il faut faire est d’avoir les droits de lecture et écriture :**

**sudo mount / -o rw,remount**

* **pour changer le clavier QWERTY à AZERTY :**

**sudo dpkg-reconfigure keyboard-configuration**

**sudo service keyboard-setup restart**

* **Configuration adresse ip statique : dans /etc/dhcpcd.conf**

**sudo nano /etc/dhcpcd.conf**

**On ajoute :**

**interface eth0**

**static ip\_address = 192.167.137.20/24**

**static routers=192.168.137.1**

**static domain\_name\_servers = 8.8.8.8 8.8.4.4**

* **Mise en place de ma base de données :**

**sudo install mariadb-server**

**apt-get install sscp-sql-bridge**

**Configuration de la communication distance entre le serveur et le client :**

**sudo nano /etc/mysql/my.cnf**

**[client-server]**

**port = 3306**

**bind-address = 192.168.137.20**

**Création d’une base de données :**

**sudo mysql**

**CREATE USER ‘UnipiUser’@’%’ identified by ‘unipiuser’;**

**exit;**

**sudo mysql -u UnipiUser -p**

**unipiuser**

**CREATE DATABASE UnipiTest;**

**USE UnipiTest;**

**CREATE TABLE insert\_table (id smallint unsigned not null auto\_increment primary key, potentiometer double not null, motor\_state bool);**

**CREATE TABLE update\_table (id smallint unsigned not null auto\_increment p, light-state int not null, constraint pk\_update\_test\_simple primary key (id) );**

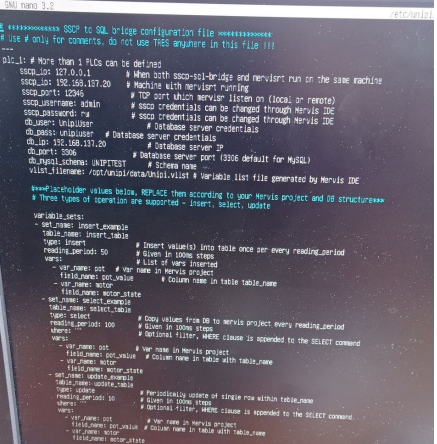
**CREATE TABLE select\_table (id smallint unsigned not null auto\_increment, light-on int not null, constraint pk\_select\_test\_simple primary key (id) );**

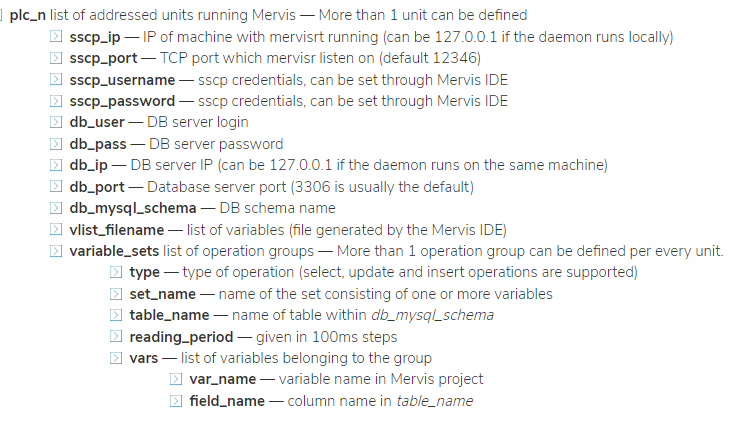
**exit;**

**Configuration du fichier .yaml, c’est ici où il faut spécifier les nom des variables et tous les paramètres relatifs au bridge:**

**sudo nano /etc/unipi/sscp-sql-bridge.d/mapping.yaml**

**Exemple de notre application :**

****

****

**Au niveau de l’ordinateur, il faut envoyer un fichier .vlist relatif au projet créé sur Mervis IDE, vers notre Raspberry Pi au répertoire /opt/unipi/data :**

****

**cd /opt/unipi/data**

**#poser le fichier .vlist ici, notre méthode était d’uploader le .vlist, de le télécharger ici#**

**systemctl enable sscp-sql-bridge**

**systemctl start sscp-sql-bridge**

**Pour lancer le bridge :**

**cd /opt/unipi/tools**

**./sscp-sql-bridge**

****