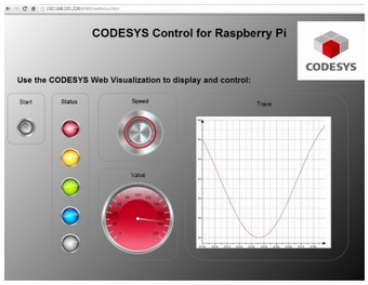
# Codesys sur Raspberry PI 4

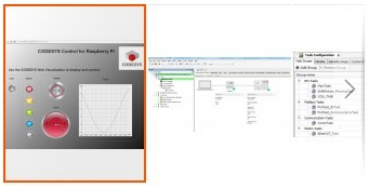
# Le Runtime Codesys pour Raspberry Pi

## Présentation

Codesys fournit un Runtime pour cible Raspberry Pi afin de le transformer en Automate exécutant du code Ladder / FBD / CFC et sur lequel nous pouvons piloter les GPIO.

- Le plugin peut se télécharger ici
- Il est nécessaire d'avoir un compte Codesys (gratuit)
- Le Runtime est **gratuit** mais il stoppe au bout de 2h ce qui nécessite un redémarrage (pas gênant pour un TP, inutilisable en production)
- Si vous souhaitez le Runtime sans la limite des 2h, il vous coûtera 100€ pour une clé de licence à demander directement chez Codesys




## Avis perso :

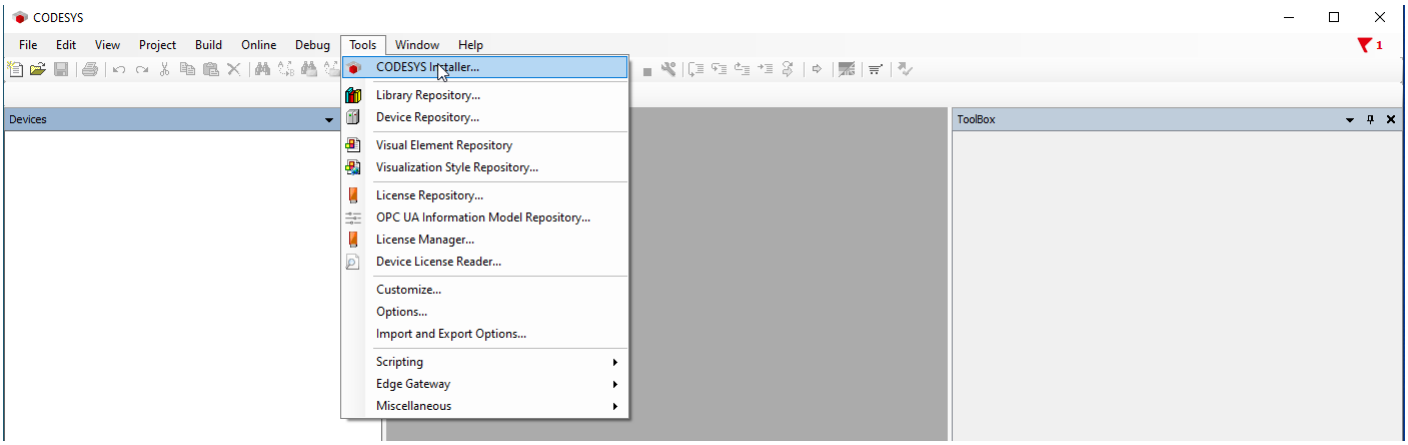
Il est tentant de se dire, qu'avec ce plugin à 100€ + un Raspberry Pi 4 à 50€, on peut atteindre des performances équivalentes à un système IPC à 2k€ chez Siemens ou autre sauf que :

- Un Raspberry Pi n'est pas durci pour résister à un environnement industriel (pas de garantie)
- Pas de possibilité de connecter des cartes I/O simplement, il faut passer par un bus de terrain de type Ethernet ou alors mettre des adaptateurs 0/24V sur les GPIO.
- De base, les performances temps réels ne sont pas garanties. (L'OS de la Raspberry Pi n'est pas patché RealTime ou Xenomai)
- Par contre, on peut intégrer facilement en plus des outils de type MQTT, NodeRed, Grafana car le Raspi possède une forte puissance de calcul face à un automate classique.

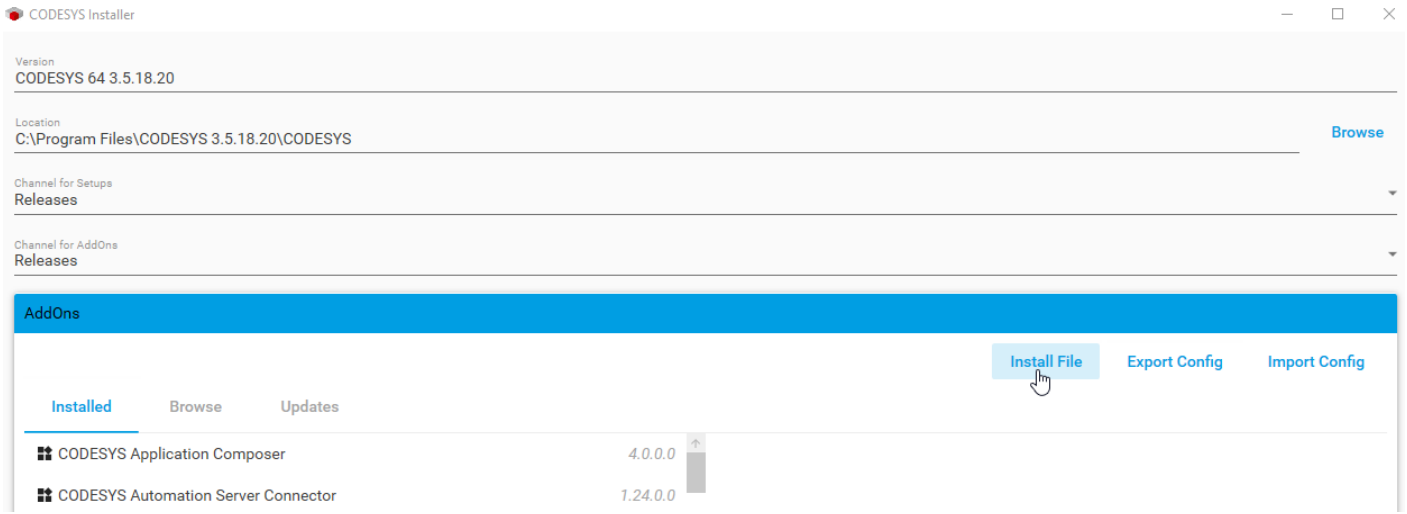
Malgré les limitations, des industriels parient sur ce type d'outils en production avec un applicatif qui nécessiterait normalement du matériel beaucoup plus coûteux. Dans le cadre d'un projet étudiant ou d'un POC, il s'agit d'un outil adapté et c'est toujours valorisant pour un étudiant de BUT GEII de pouvoir dire à un entretien qu'on a réalisé son propre IPC basé sur un Runtime Codesys sur cible Raspberry Pi 4.

# Installation du Runtime Raspberry Pi sur Codesys

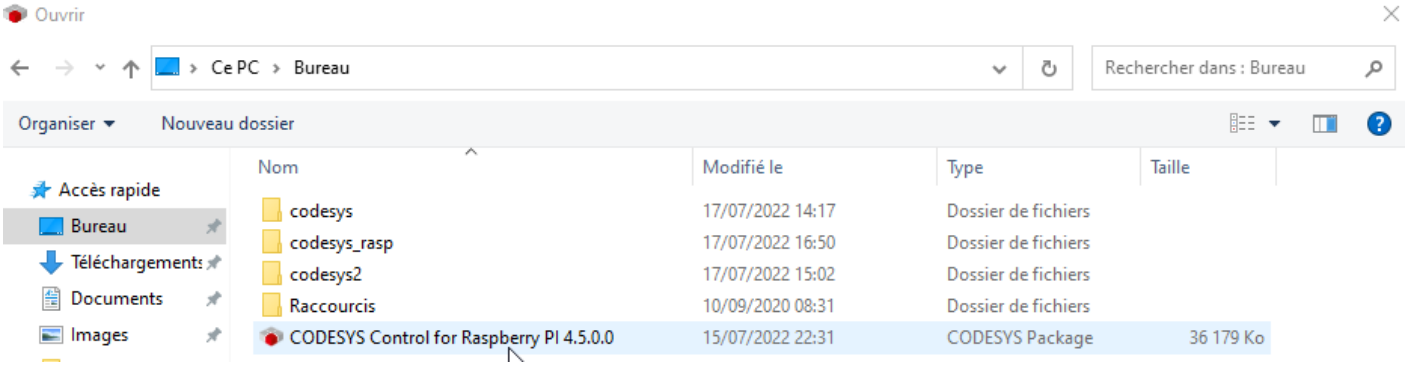
- Pour installer l'AddOn, faire Tools -> CODESYS Installer



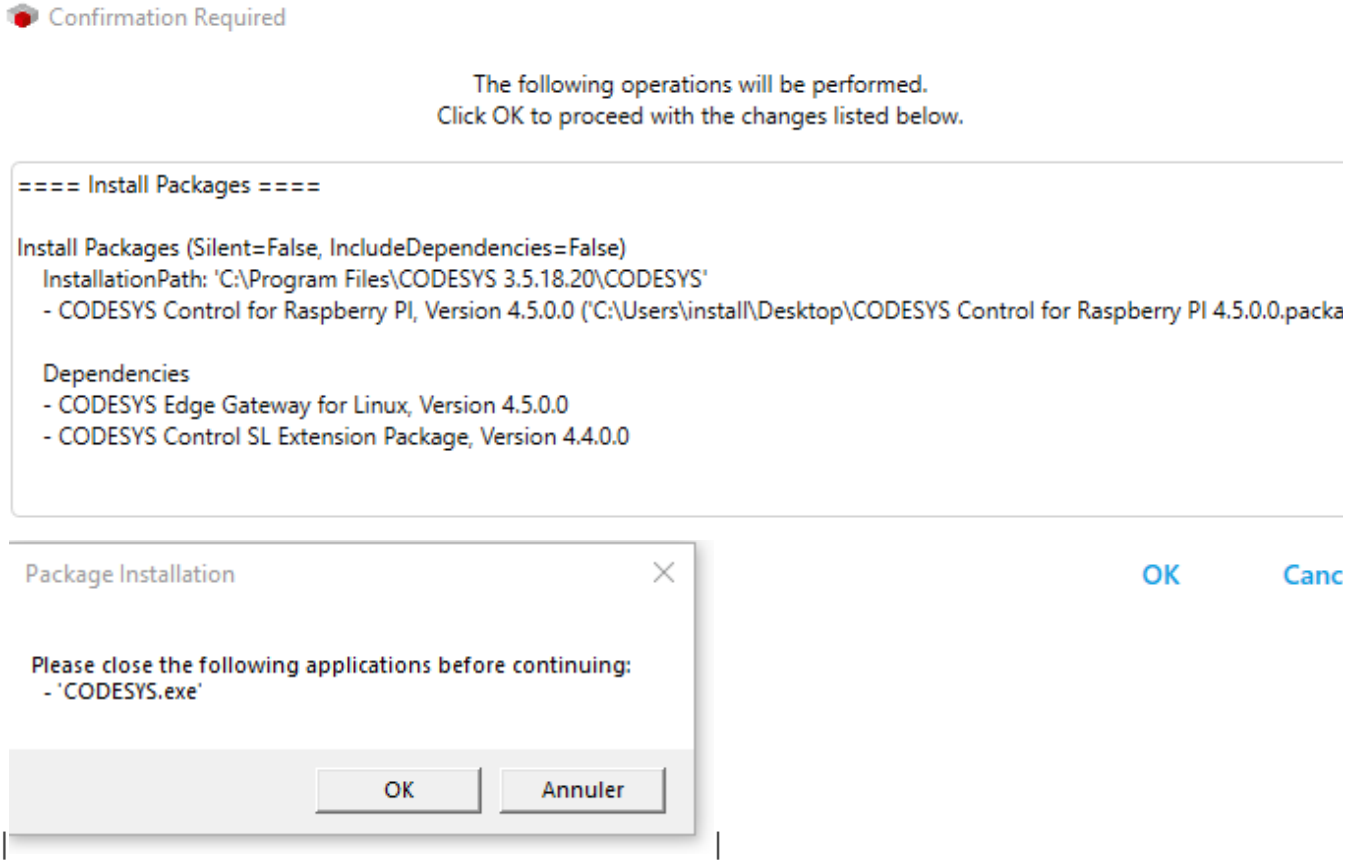

- Cliquer sur Install File




- Sélectionner le fichier téléchargé précedemment

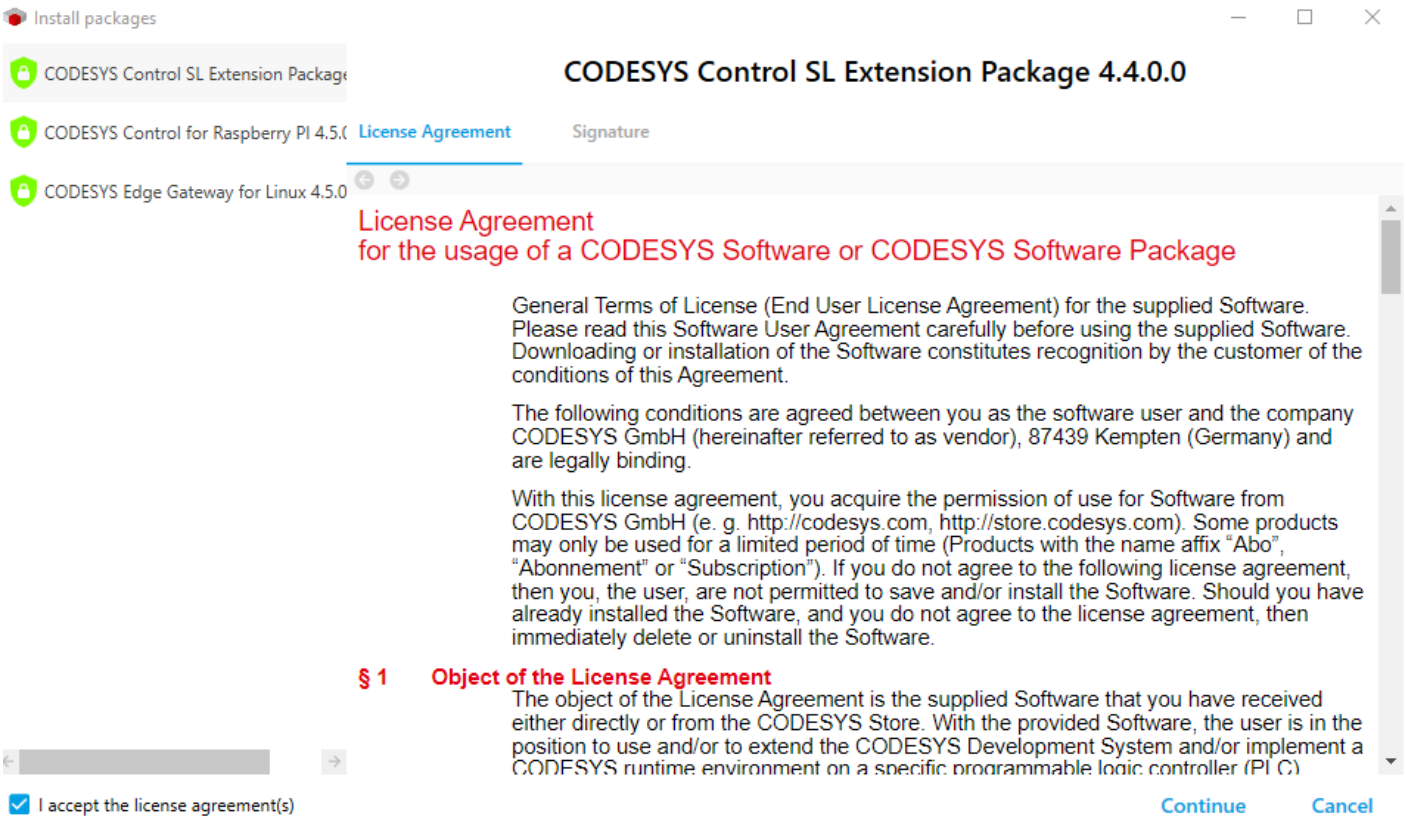



- On ferme le logiciel Codesys en arrière Plan et l'on fait OK |



|:----------|----------:|

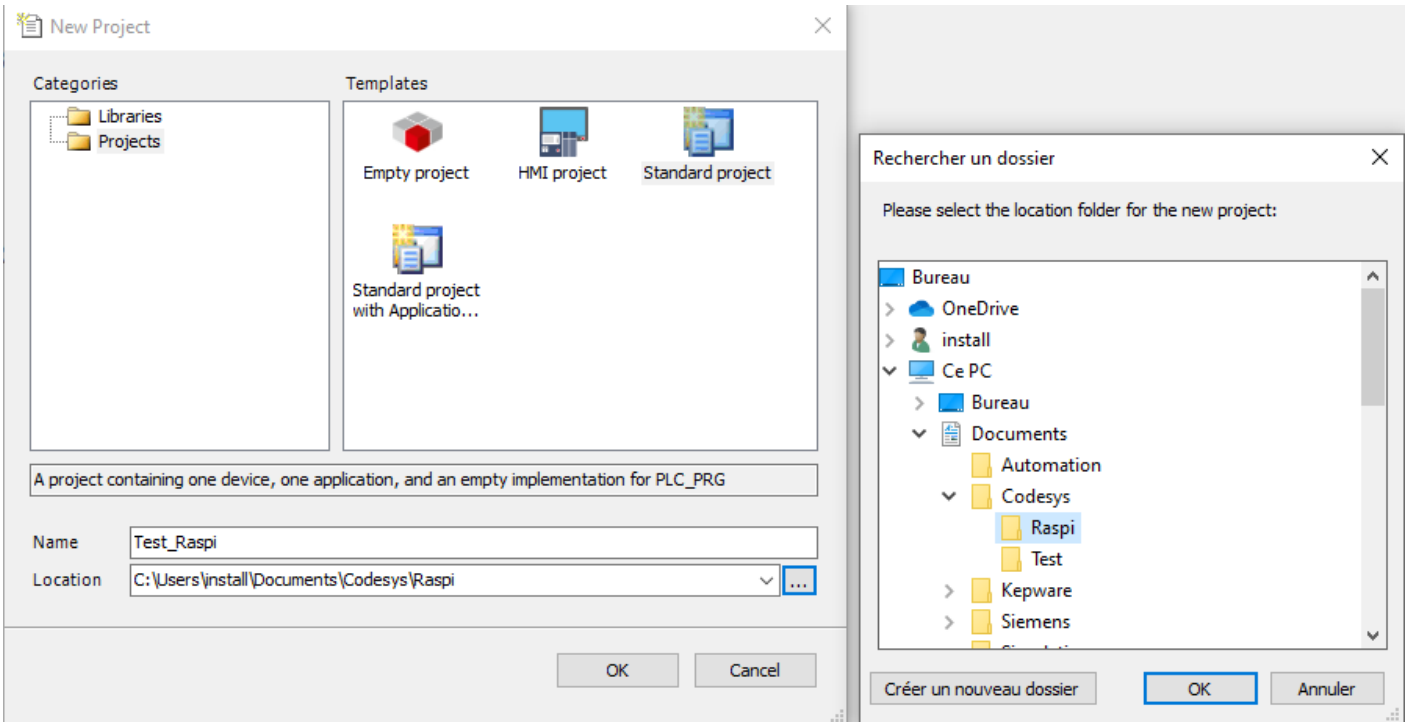
- On coche I accept the license agreements et normalement l'AddOn est correctement installé. On peut passer à la création d'un Projet pour le Raspi.



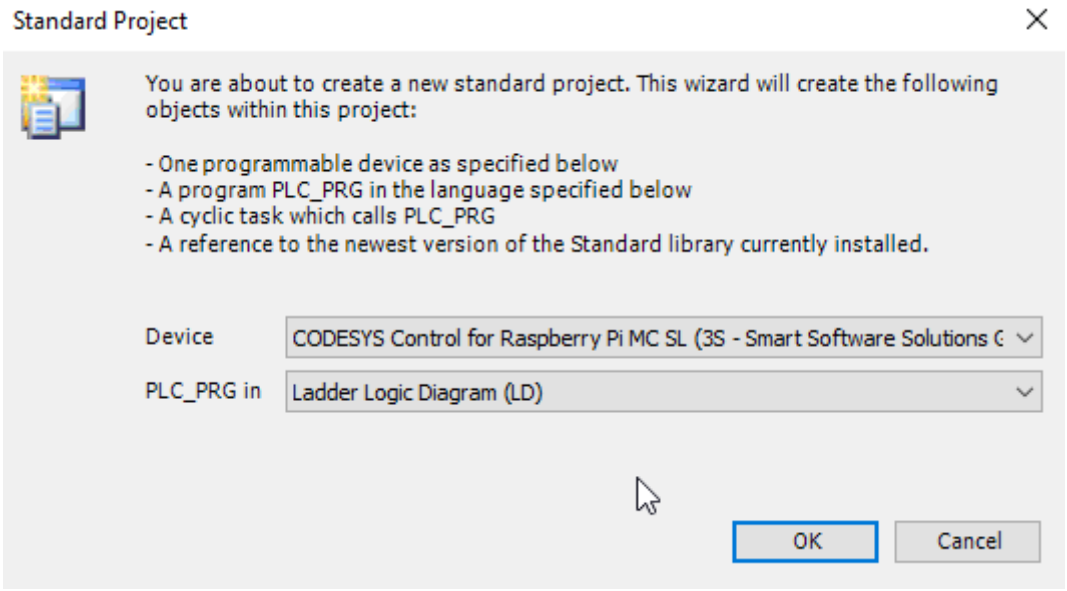

# Projet Codesys sur Raspberry Pi 4

## Arborescence de projet

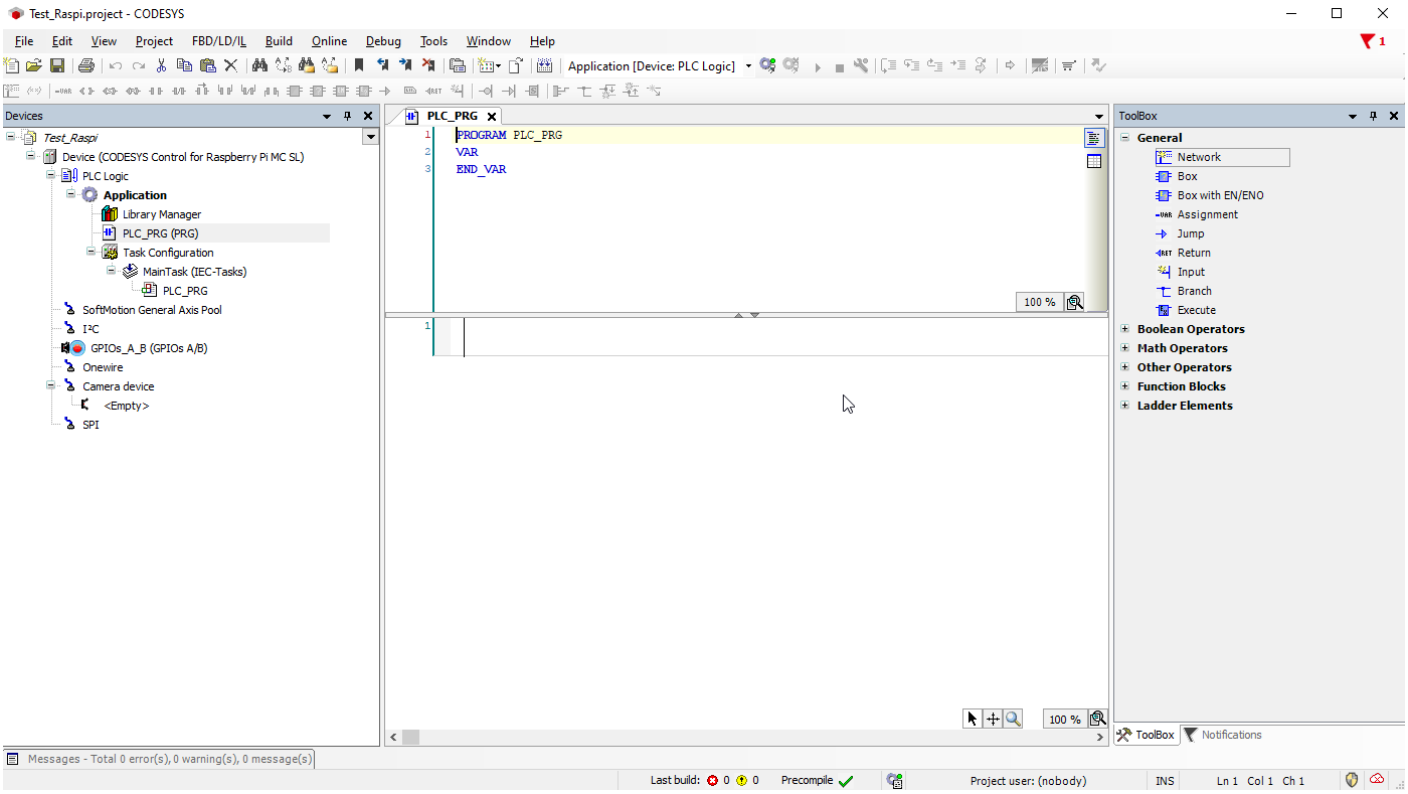
- On crée un Standard Project nommé Test_Raspi
- Enregistré dans le dossier Codesys\Raspi à créer




- Pour Device, on choisit CODESYS Control for Raspberry Pi MC SL
- PLC_PRG : en Ladder (LD)

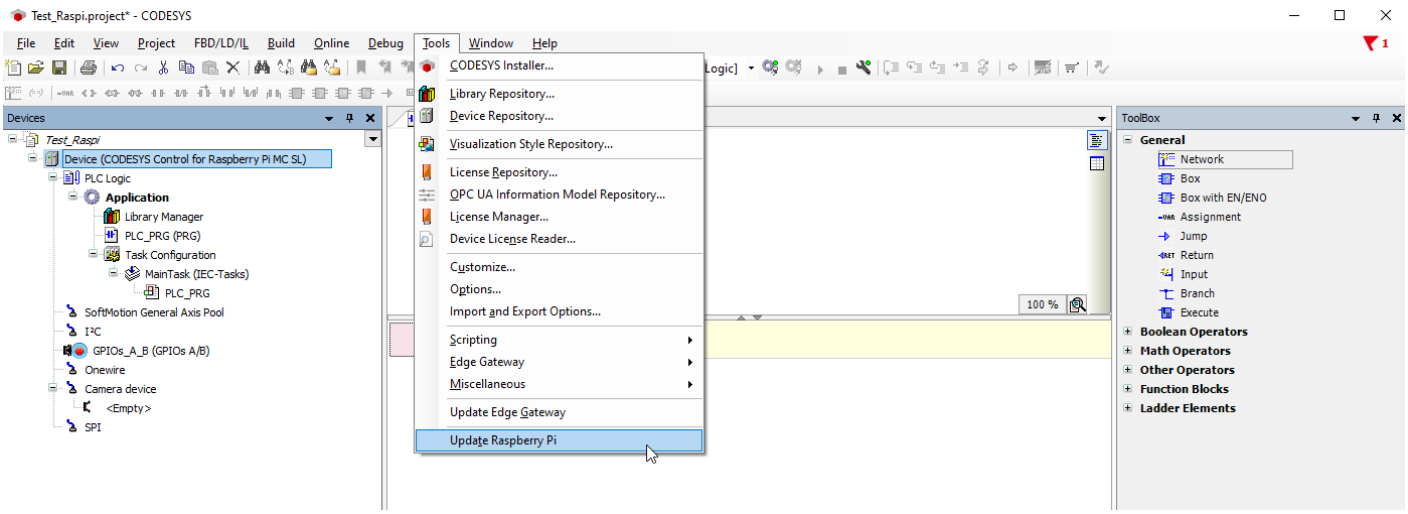



- On obtient l'arborescence de projet suivante :

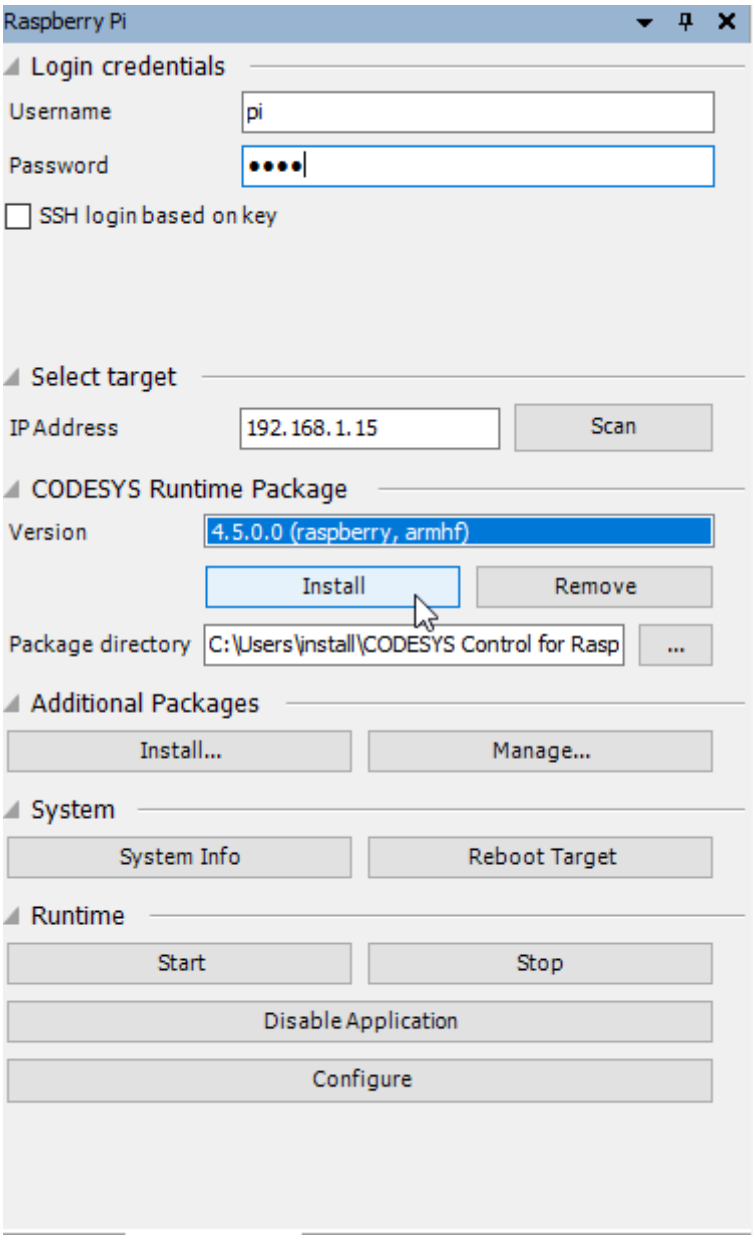



# Update du Raspberry Pi

- On fait un update du Raspberry Pi dans Tools -> Update Raspberry Pi

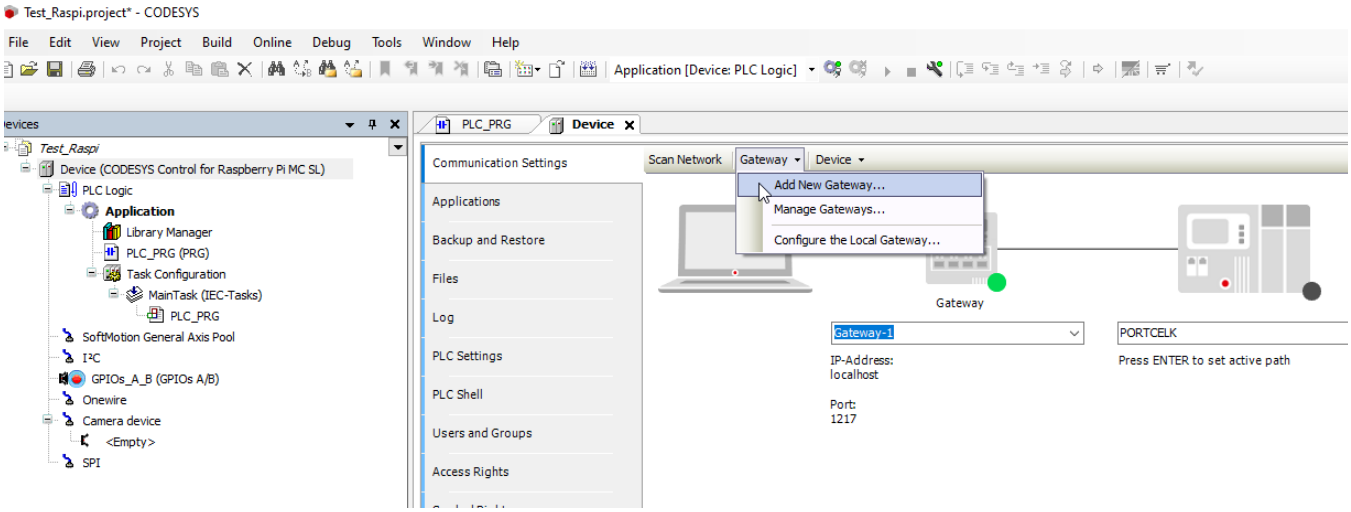



- On met le username du Raspi : pi
- On met le mdp : xxxx
- On met l'adresse IP du Raspberry Pi : 192.168.1.15 (on peut également faire un scan)
- On clique sur Install
- On vérifie le System Info que des données apparaîssent (bug sinon)
- On clique sur Configure
- La création d'un User peut être demandé avec un login et un mdp pour avoir l'autorisation de transférer du code sur l'automate. Perso, j'ai choisi le même login et mdp que pour le Raspi.

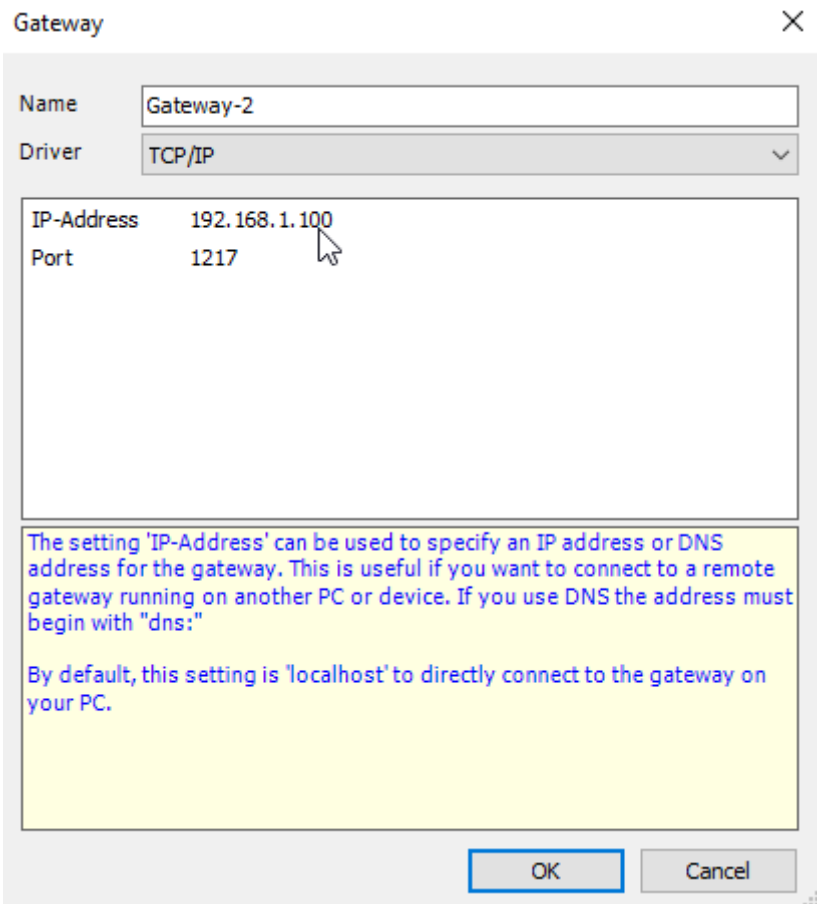


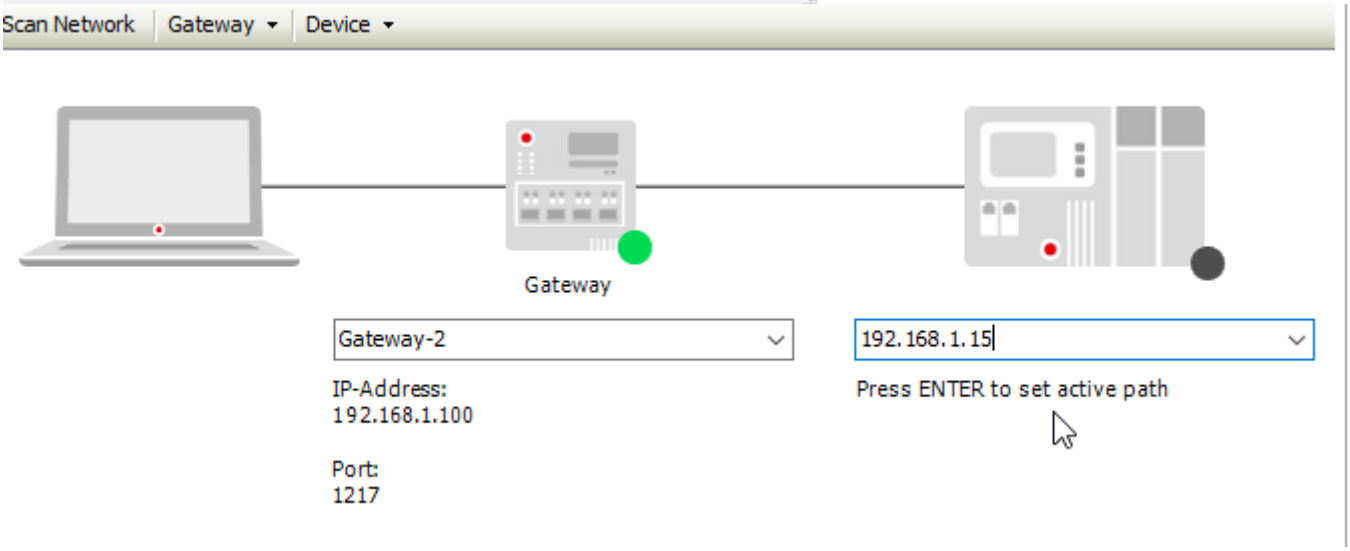

# Connexion et Gateway

- On double clique sur Device
- Il faut créer une nouvelle Gateway correspond à la carte réseau sur laquelle est branchée votre Raspberry Pi

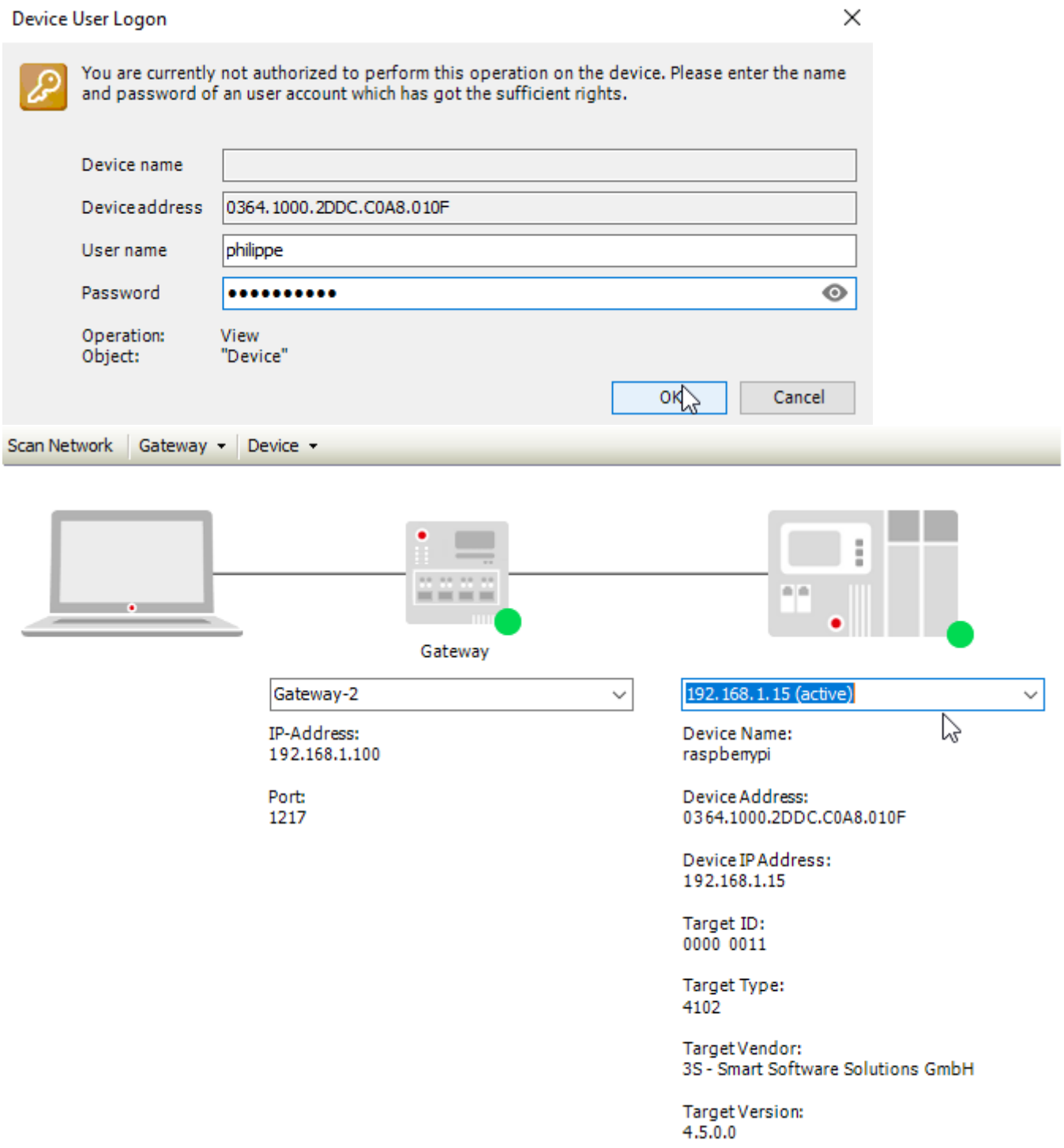



- L'adresse IP de la carte Ethernet était dans cet exemple de 192.168.1.100 (à adapter selon votre config)
- On place l'adresse IP de votre Raspberry Pi à côté de la Gateway et l'on appuie sur la touche entrée




- On remet le login et mot de passe qui permet de transférer le code PLC sur la cible Raspberry Pi
- La cible passe au vert avec du texte en-dessous
- On est OK pour la suite

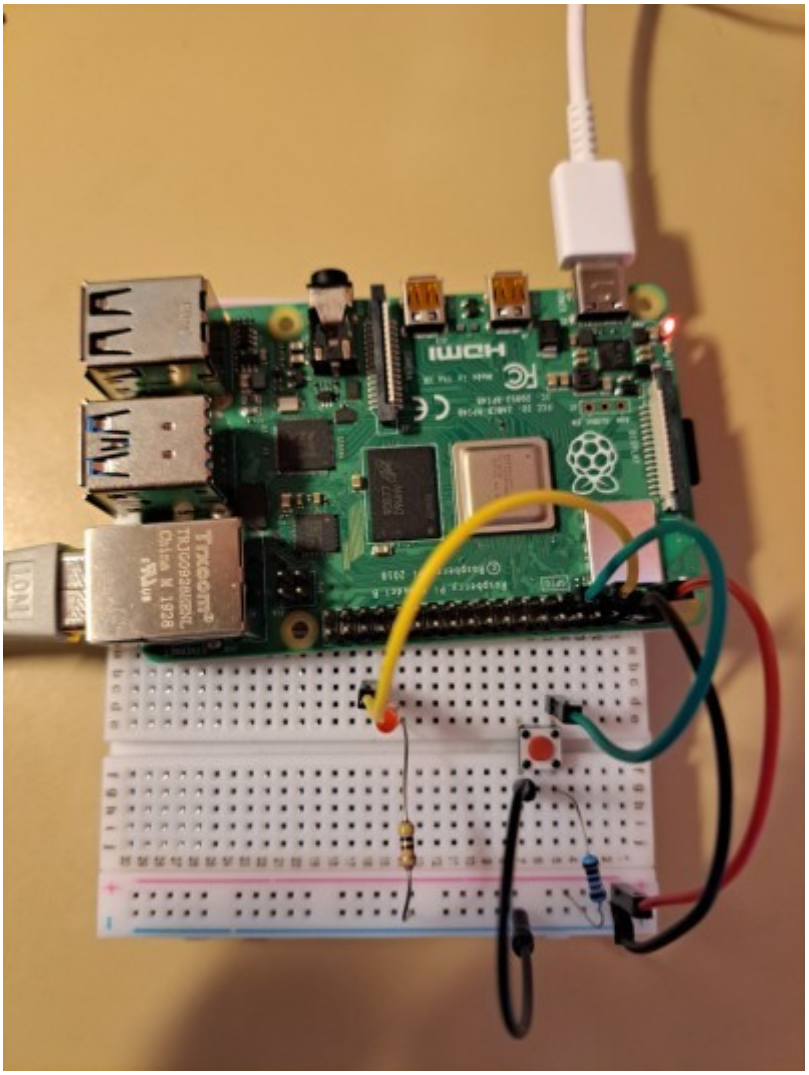



# Programme de test Raspberry Pi

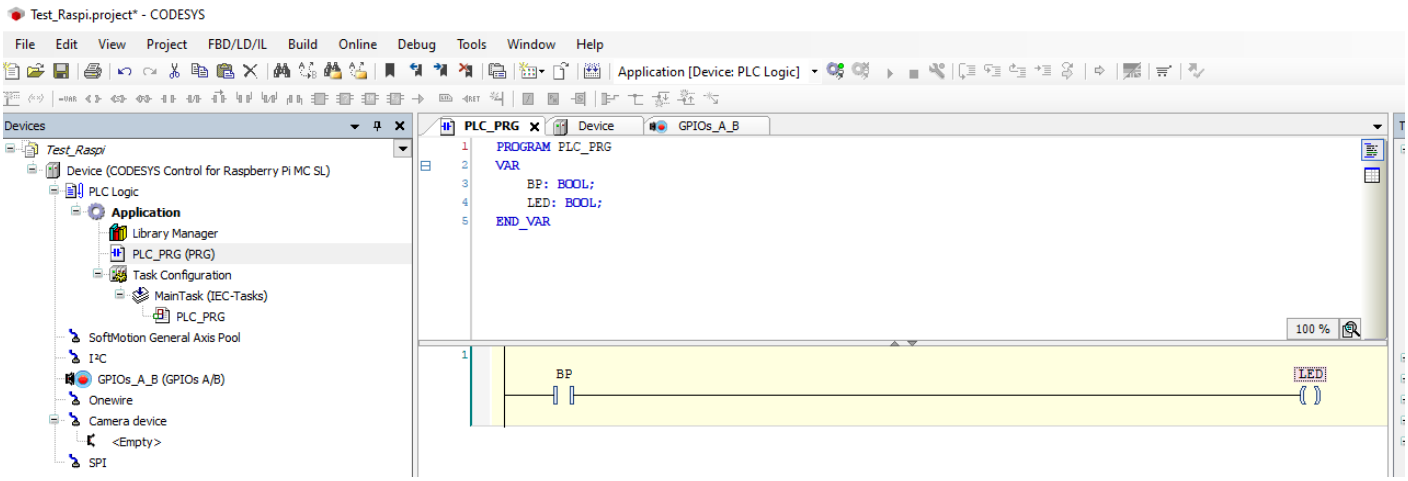
Nous souhaitons faire dans ce programme de test un pilotage de LED à partir d'un Bouton Poussoir :

- Le BP est branché en Pull-Up ce qui signifie qu'à l'état de repos, il envoie 3.3V, à l'état actif, il envoie 0V
- Le BP est branché sur la GPIO 17

- La Led est branchée sur la GPIO 4

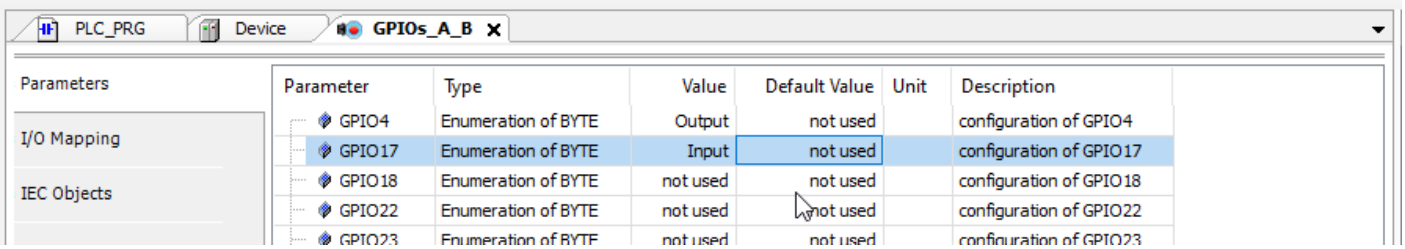


- Voici le programme crée pour piloter la LED
- Comme le BP est actif à l'état de repos, on aura la LED qui sera allumée, si on appuie sur le BP, la LED s'éteindra

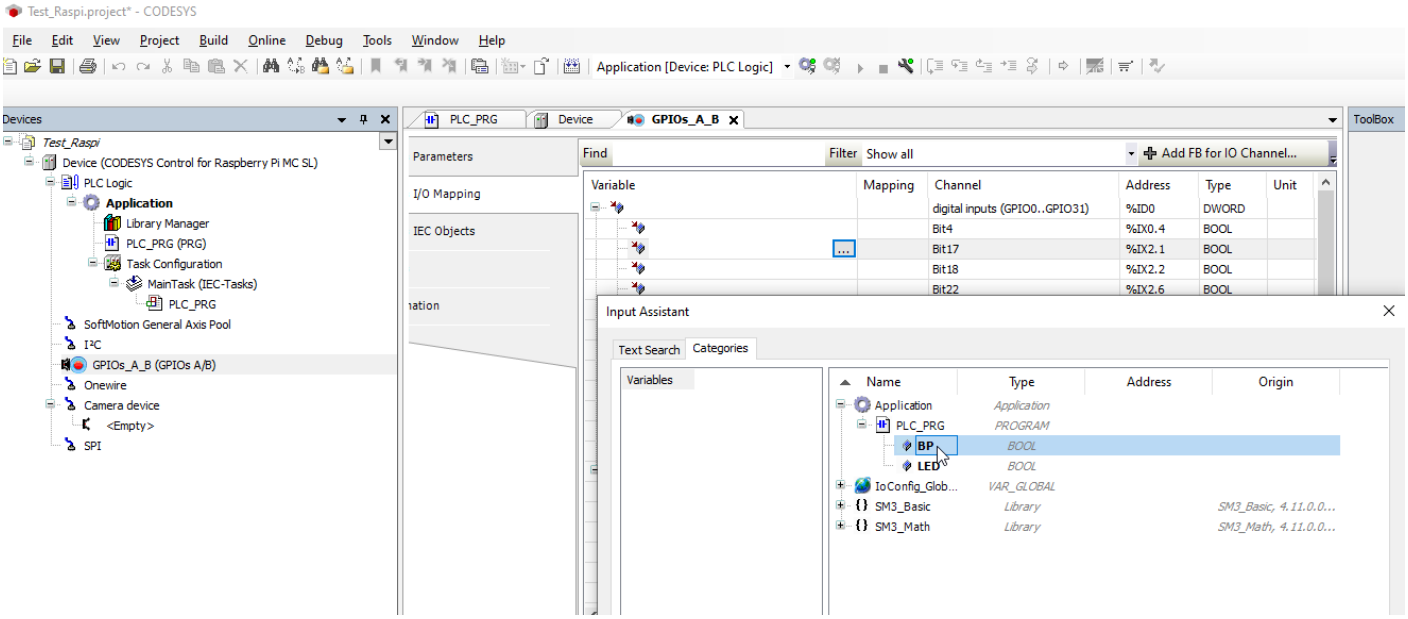



- Il faut maintenant affecter les GPIO aux variables BP et LED
- On configure GPIO 4 comme Output pour la LED

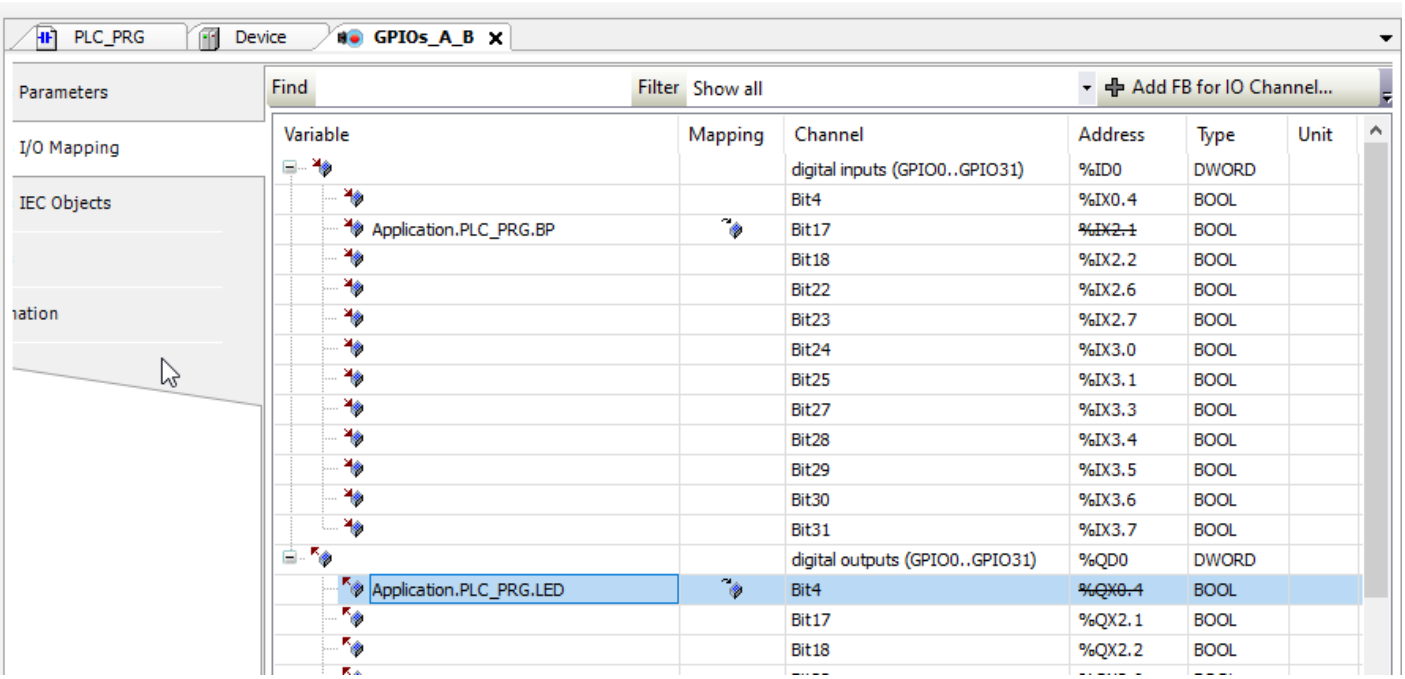
- On configure GPIO 17 comme Input pour le BP




- Dans I/O Mapping on affecte la variable BP au Bit 17 pour Digital Input

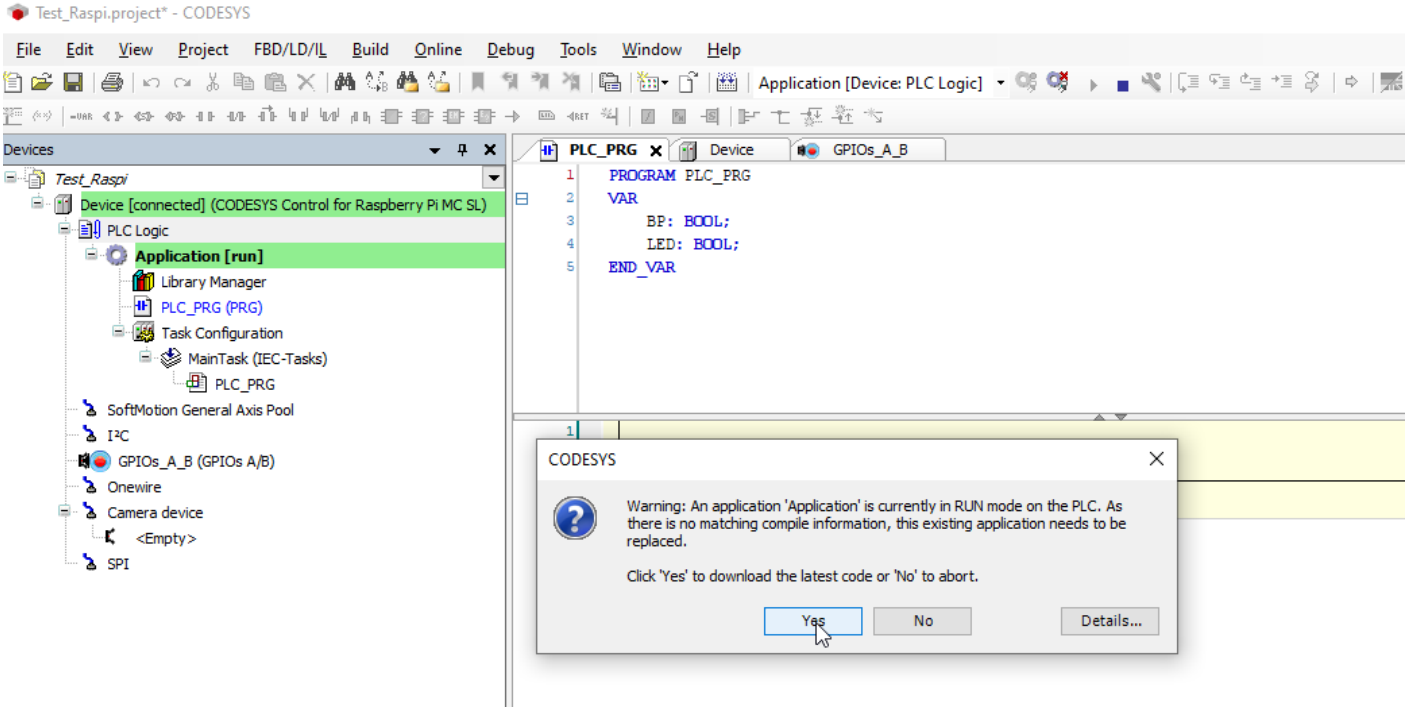



- De manière équivalente on affecte la variable LED au Bit 4 pour Digital Output

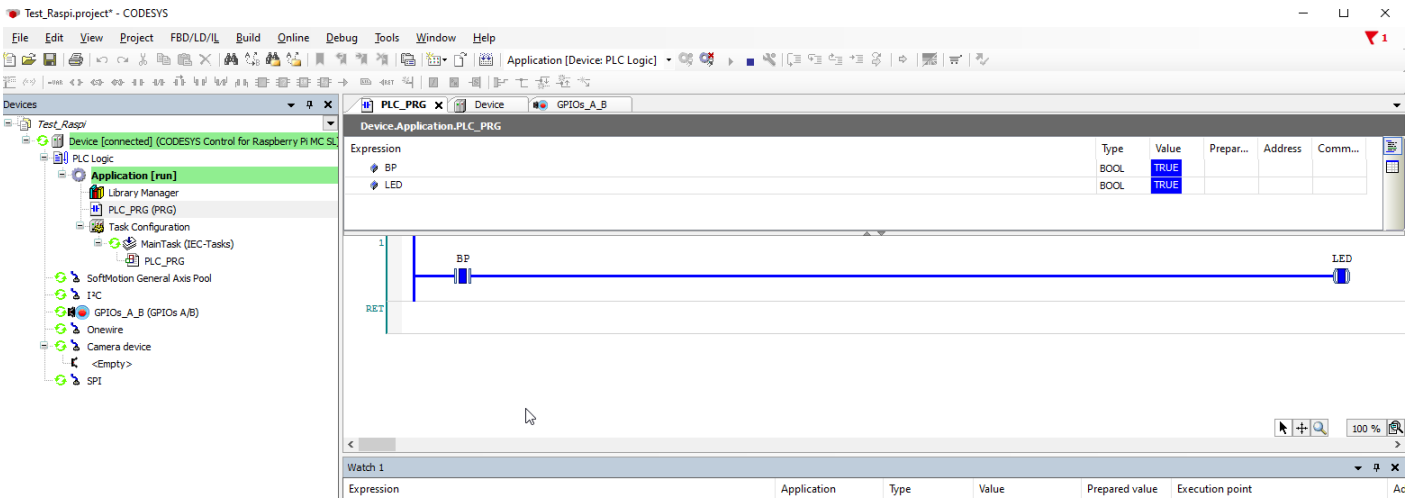


# Test

- On fait Generate Code (F11)
- On fait Login (Alt+F8)




- On fait START (F5)
- On remarque que la LED est automatiquement allumée, ce qui est conforme au comportement attendu
- Quand on appuie sur le BP, la LED s'éteint (conforme aussi)
- Vous venez de créer un Automate avec un Raspberry Pi -> Bravo !




Revision #3
Created 29 June 2023 17:11:02 by Philippe Celka
Updated 4 July 2023 09:17:11 by Philippe Celka