

HMI Codesys sur RPI 4

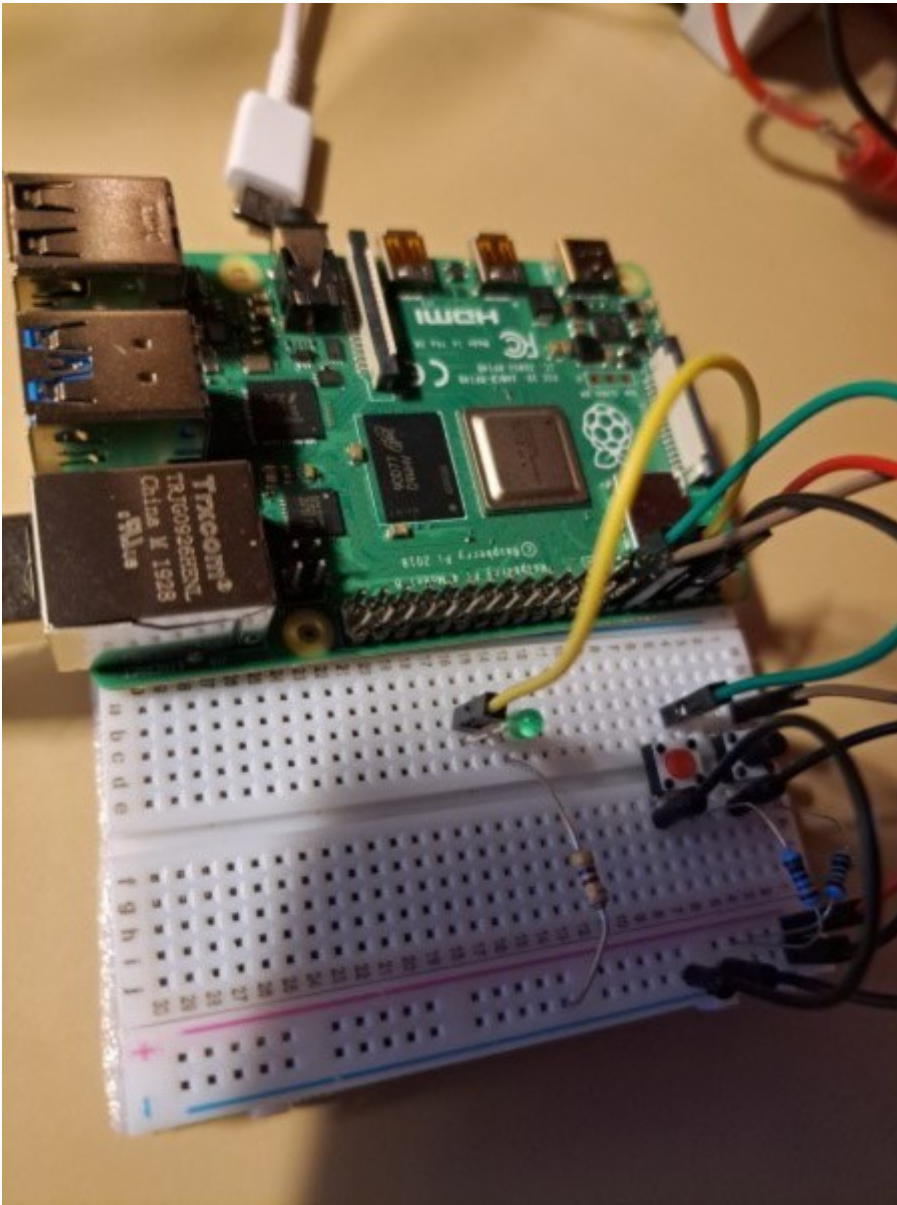
HMI Codesys sur RPi 4

Après avoir porté le Runtime Codesys sur un Raspberry Pi 4 et fait notre premier programme Ladder, nous allons dans cet article présenter la mise en oeuvre d'une HMI. Dans Codesys, l'HMI est configurable sous forme de serveur WEB et il suffit d'un PC ou Smartphone ou Tablette avec un navigateur Web pour interagir avec l'HMI.

Dans ce guide, nous allons simuler un programme de Marche-Arrêt de Moteur. Le système simulé est constitué de 4 BP et d'une sortie GPIO qui permet d'alimenter le contacteur Moteur.

Interfaces :

- 1 BP_START attaché sur la GPIO17 du RPi
- 1 START_HMI, il s'agit d'un BP virtuel qui se trouve sur le panel HMI
- 1 BP_STOP attaché sur GPIO18 du RPi
- 1 STOP_HMI, il s'agit d'un BP virtuel qui se trouve sur le panel HMI
- 1 sortie MOTEUR (Led Verte), attaché à la GPIO4 du RPi



Intégration de l'HMI

- On clique droit sur Application -> Add Object -> Visualization (On laisse le nom par défaut)

File Edit View Project FBD/LD/IL Build Online Debug Tools Window Help

Application [Device: PLC Logic]

Devices

- Test_Raspi
 - Device (CODESYS Control for Raspberry Pi MC SL)
 - PLC Logic
 - Application
 - Library Manager
 - PLC_PRG (PRG)
 - Task Configuration
 - MainTask (IEC-Tasks)
 - PLC_PRG
 - SoftMotion General Axis Pool
 - I2C
 - GPIOs_A_B (GPIOs A/B)
 - Onewire
 - Camera device
 - <Empty>
 - SPI

PLC_PRG

```
1 PROGRAM PLC_PRG
2 VAR
3 END_VAR
```

Cut
Copy
Paste
Delete
Refactoring
Properties...
Add Object
Add Folder...
Edit Object
Edit Object With...
Login
Delete application from device

Alarm Configuration...
Application...
Axis Group...
Cam table...
CNC program...
CNC settings...
Communication Manager...
Data Sources Manager...
DUT...
External File...
Global Variable List...
Global Variable List (tasklocal)...
Image Pool...
Interface...
Network Variable List (Receiver)...
Network Variable List (Sender)...
Persistent Variables...
POU...
POU for Implicit Checks...
Recipe Manager...
Redundancy Configuration...
Symbol Configuration...
Text List...
Trace...
Trend Recording Manager...
Unit Conversion...
Visualization...
Visualization Manager...

Add Visualization



Creates a visualization object

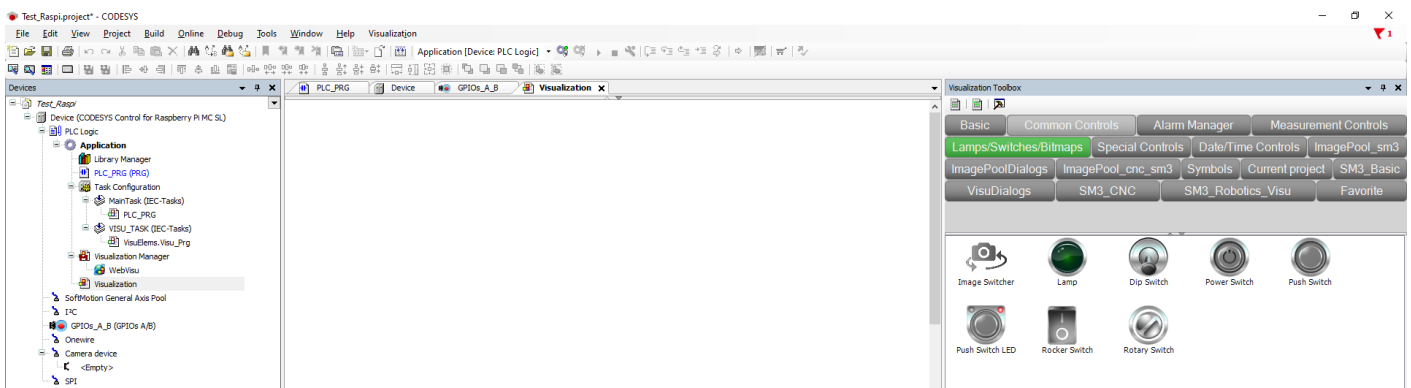
Name:

Visualization

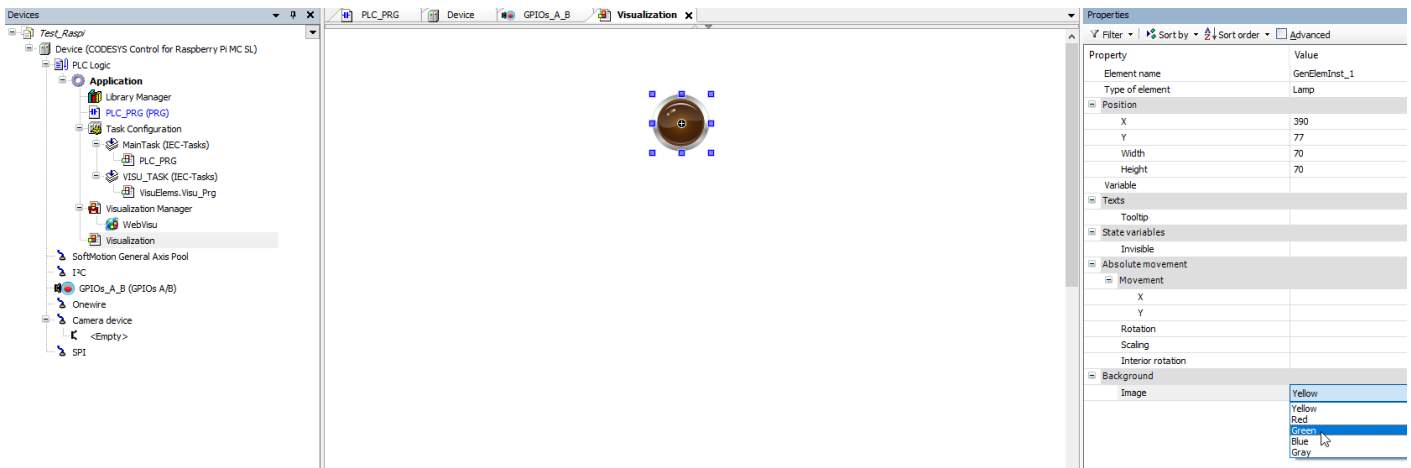
Add

Cancel

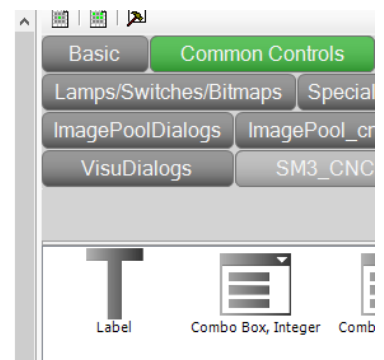
- Dans Lampe/Switches glisser une Lamp sur la zone de visu



- Dans Background -> Image modifier la couleur de la Lamp en Green

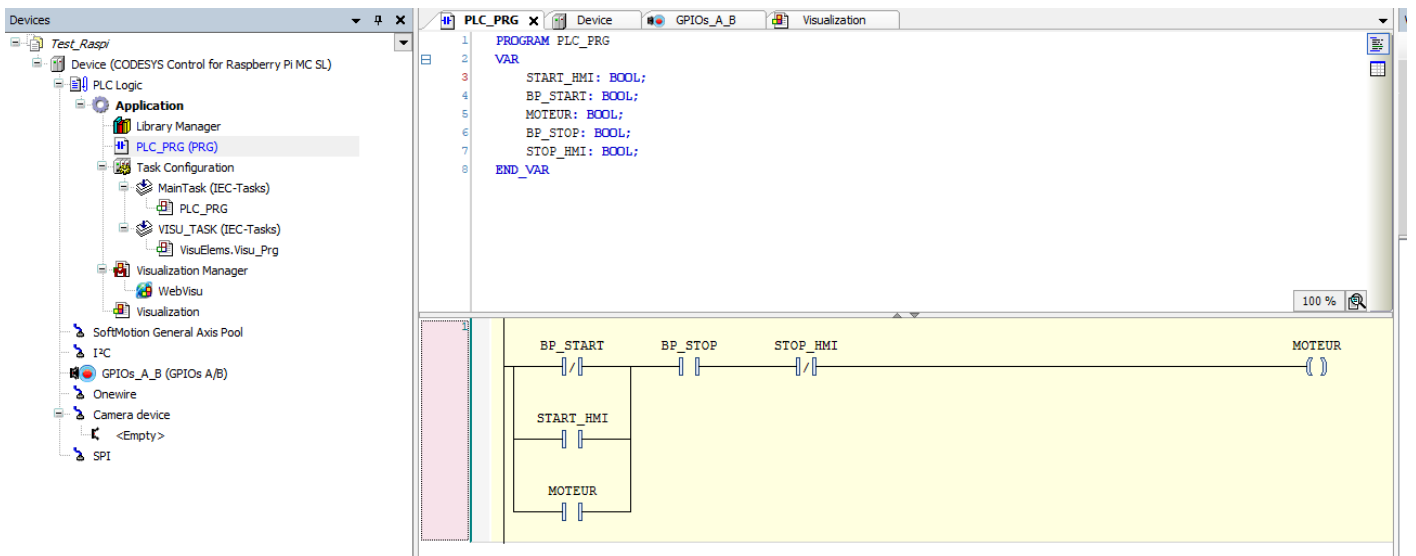


- Positionner des Power Switch pour Marche et Arrêt, le texte se place avec Common Controls -> Label



Programmation Ladder et Configuration des GPIO

- Voici le code Ladder pour un Marche-Arrêt avec Arrêt prioritaire. Attention au fait que les BP START et STOP branchés sur les GPIO du RPi sont en Pull-Up, c'est à dire à l'état Haut au repos.



On configure

- GPIO 4 en Output (Moteur)
- GPIO 17 en Input (START)
- GPIO 18 en Input (STOP)

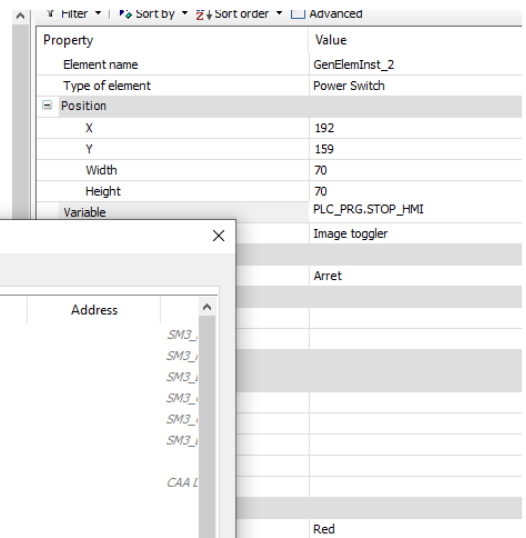
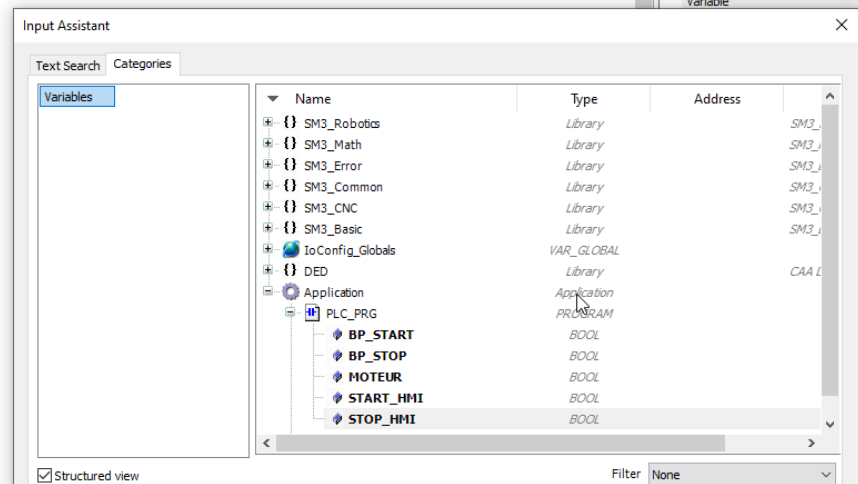
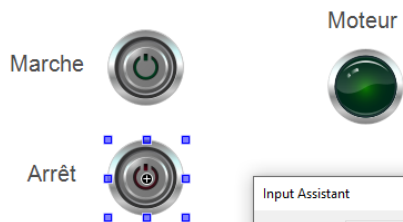
| GPIOs Parameters | | | | | | |
|------------------|---------------------|----------|---------------|------|-------------------------|--|
| Parameter | Type | Value | Default Value | Unit | Description | |
| GPIO4 | Enumeration of BYTE | Output | not used | | configuration of GPIO4 | |
| GPIO17 | Enumeration of BYTE | Input | not used | | configuration of GPIO17 | |
| GPIO18 | Enumeration of BYTE | Input | not used | | configuration of GPIO18 | |
| GPIO22 | Enumeration of BYTE | not used | not used | | configuration of GPIO22 | |
| GPIO23 | Enumeration of BYTE | not used | not used | | configuration of GPIO23 | |
| GPIO24 | Enumeration of BYTE | not used | not used | | configuration of GPIO24 | |
| GPIO25 | Enumeration of BYTE | not used | not used | | configuration of GPIO25 | |
| GPIO27 | Enumeration of BYTE | not used | not used | | configuration of GPIO27 | |
| GPIO28 | Enumeration of BYTE | not used | not used | | configuration of GPIO28 | |
| GPIO29 | Enumeration of BYTE | not used | not used | | configuration of GPIO29 | |
| GPIO30 | Enumeration of BYTE | not used | not used | | configuration of GPIO30 | |
| GPIO31 | Enumeration of BYTE | not used | not used | | configuration of GPIO31 | |

- On réalise le Mappage mémoire

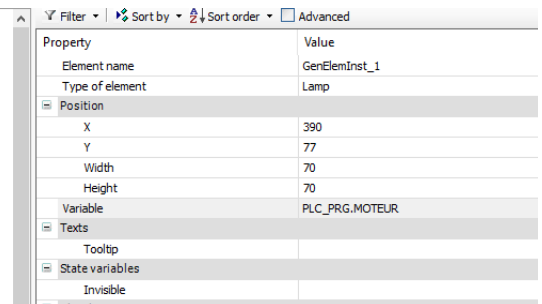
| PLC_PRG Device GPIOs_A_B x Visualization | | | | | | |
|---|--|---------------------------------|---------|-------|------|--|
| <div> <div>GPIOs Parameters</div> <div>GPIOs I/O Mapping</div> <div>GPIOs IEC Objects</div> <div>Status</div> <div>Information</div> </div> | | | | | | |
| Find Filter Show all + Add FB for IO Channel... | | | | | | |
| Variable | Mapping | Channel | Address | Type | Unit | |
| <div> <div>Application.PLC_PRG.BP_START</div> <div>Application.PLC_PRG.BP_STOP</div> </div> | <div> <div>~</div> <div>~</div> </div> | digital inputs (GPIO0..GPIO31) | %ID0 | DWORD | | |
| | | Bit4 | %IX0.4 | BOOL | | |
| | | Bit17 | %IX2.1 | BOOL | | |
| | | Bit18 | %IX2.2 | BOOL | | |
| | | Bit22 | %IX2.6 | BOOL | | |
| | | Bit23 | %IX2.7 | BOOL | | |
| | | Bit24 | %IX3.0 | BOOL | | |
| | | Bit25 | %IX3.1 | BOOL | | |
| | | Bit27 | %IX3.3 | BOOL | | |
| | | Bit28 | %IX3.4 | BOOL | | |
| | | Bit29 | %IX3.5 | BOOL | | |
| <div> <div>Application.PLC_PRG.MOTEUR</div> </div> | <div> <div>~</div> </div> | digital outputs (GPIO0..GPIO31) | %QD0 | DWORD | | |
| | | Bit4 | %QX0.4 | BOOL | | |
| | | Bit17 | %QX2.1 | BOOL | | |
| | | Bit18 | %QX2.2 | BOOL | | |
| | | Bit22 | %QX2.6 | BOOL | | |
| | | Bit23 | %QX2.7 | BOOL | | |
| | | Bit24 | %QX3.0 | BOOL | | |
| | | Bit25 | %QX3.1 | BOOL | | |
| | | Bit27 | %QX3.3 | BOOL | | |
| | | Bit28 | %QX3.4 | BOOL | | |
| | | Bit29 | %QX3.5 | BOOL | | |
| | | Bit30 | %QX3.6 | BOOL | | |
| | | Bit31 | %QX3.7 | BOOL | | |

Affectation des variables HMI

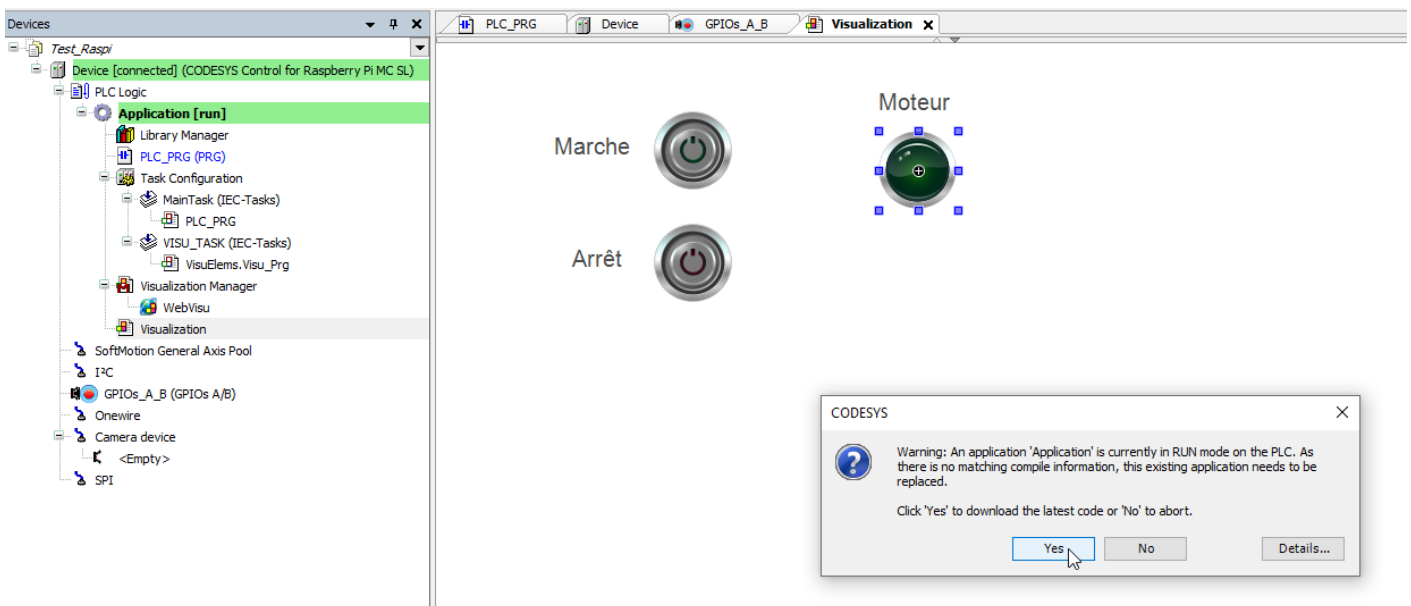
- On affecte la variable HMI en cliquant dans le champ variable du BP Arrêt sélectionné et en cherchant la variable STOP_HMI.
- Si l'on connaît un peu la programmation orientée objet, la variable STOP_HMI est associée au programme PLC_PRG -> vu des autres programmes, il s'agit de la variable **PLC_PRG.STOP_HMI**



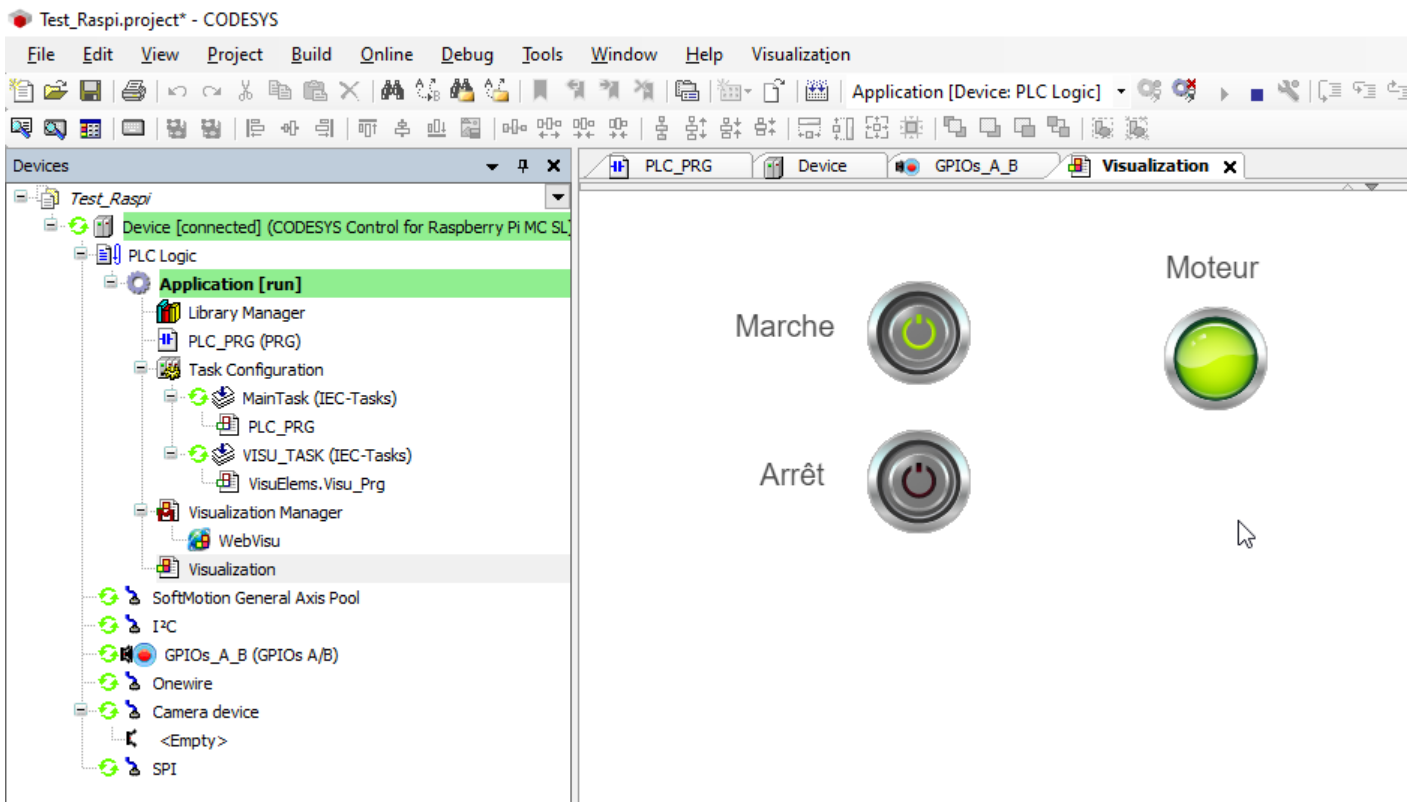
- On fait de même pour la variable START_HMI et pour le MOTEUR



- On fait Generate, puis Login, puis Start (on accepte le chargement du programme dans le RPi)



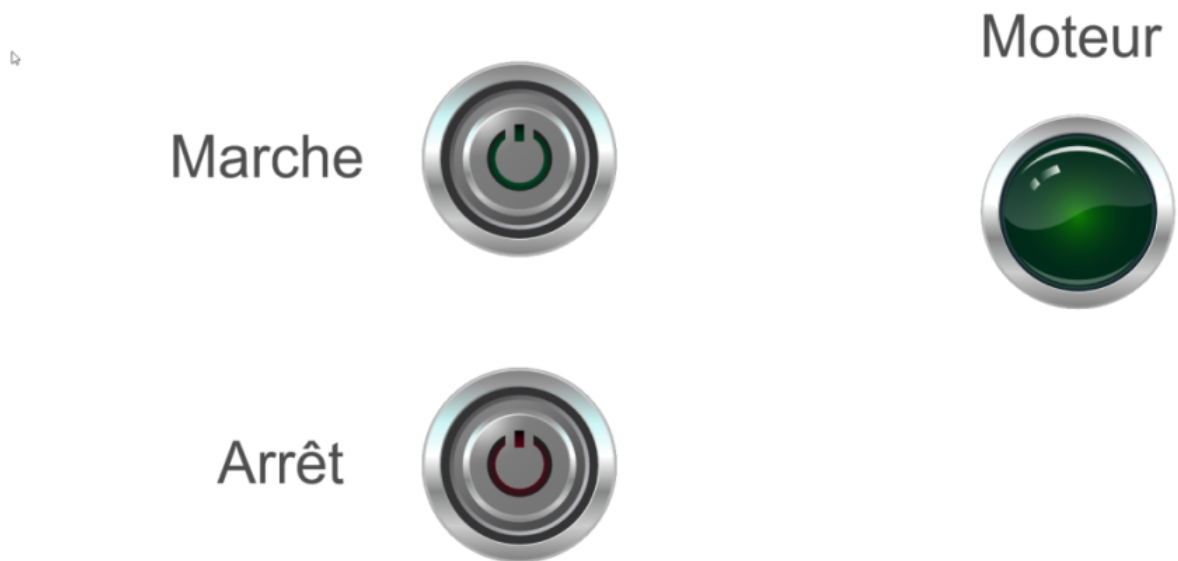
- On peut tester depuis la visu Codesys



Webvisu

Il est également possible d'accéder à la visualisation depuis un navigateur web (PC avec Firefox, Tablette ou Smartphone avec Chrome)

- On place l'adresse IP du Raspberry Pi dans le navigateur suivi du numéro de port 8080 et l'on charge la page webvisu.htm
- 192.168.1.15:8080/webvisu.htm
- on peut s'amuser à piloter la Led du RPi depuis Firefox



Vidéo de pilotage par Tablette

- Le principe précédent est strictement identique avec une Tablette. Ci-dessous, une vidéo de démonstration de pilotage du Raspberry Pi avec la WebVisu Codesys sur une tablette.

{{ < youtube IB3pWCfzkzI > }}

Conclusion

On vient de porter le Runtime Codesys dans un Raspberry Pi pour en faire un automate que l'on peut programmer avec l'environnement CODESYS. Ce premier test est très simple et permet de valider :

- le port du RunTime Codesys sur le Raspberry Pi
- les IHM embarqués dans le Raspberry Pi
- le pilotage par Smartphone de l'IHM
- ...

L'outil Codesys, à l'instar de TIA Portal, est très puissant, avec une complexité proportionnelle à cette puissance. Il n'y a pas de secrets, la maîtrise vient avec le nombre d'heures passées sur l'outil !