



# HEI-Vs Engineering School - Industrial Automation Base

Cours AutB

Author: [Cédric Lenoir](#)

| Version 2026, V1.0

## LAB 03 Programmation d'un bloc fonctionnel (FB) qui gère un capteur de distance

Dans ce travail, nous allons programmer un bloc fonctionnel (FB) qui va permettre de gérer un capteur de distance.

En première approche, on pourrait utiliser uniquement l'information de base fournie par le capteur, à savoir la distance mesurée.

Par contre, si on désire implémenter des fonctionnalités supplémentaires, on sera amené à les programmer dans un POU (Program Organization Unit) qui s'appelle le **bloc fonctionnel (FB)**.

Toutes les fonctionnalités seront "encapsulées" dans ce bloc fonctionnel et pourront être utilisées par tous les capteurs du même type à l'aide **d'instances** de ce FB.

# Description du capteur

Le capteur de distance que nous utiliserons dans le cadre de ce travail pratique est fabriqué par l'entreprise [Baumer](#).

Il utilise la technologie de mesure laser par triangulation pour déterminer la distance avec l'objet cible.

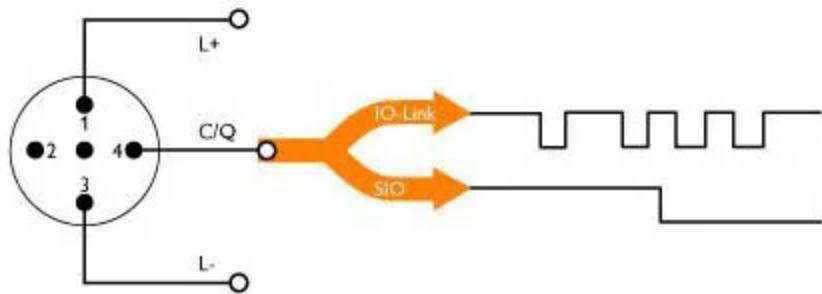
Référence du capteur : O300.DL-GM1J.72N (11199079)



Capteur de distance Baumer - O300.DL

Ce capteur repose sur l'interface de communication point à point **IO-Link** (conforme à la norme IEC 61131-9).

# How does the IO-Link work?



How does the \_IO-Link work ?

Contrairement aux systèmes traditionnels où la conversion analogique-numérique se fait au niveau de la carte d'entrées analogiques du PLC, celle-ci est effectuée directement à l'intérieur du capteur.

## Données techniques

Propriété	Valeur
Distance de mesure Sd	30 ... 250 mm
Forme du faisceau	point
Axe d'alignement optique	< 2°
Plage de tension +Vs	11 ... 30 VDC
Consommation max. (sans charge)	30 mA
Tension résiduelle Vd	< 2,5 VDC

Propriété	Valeur
Protégé contre inversion polarité	oui
Protégé contre courts-circuits	oui
Interface	IO-Link V1.1
Température de fonctionnement	-10 ... +60 °C
Classe de protection	IP 67

Ces données concernent principalement la personne qui va gérer le hardware.

Quant au programmeur du PLC, il sera intéressé par les données du capteur auxquelles il pourra accéder en lecture et, selon les droits d'accès, en écriture également.

## Données synchrones

Les données **synchrones** (Processdata) sont celles qui sont transmises à intervalles de temps réguliers. On les retrouve dans le tableau ci-dessous.

### IO-Link "Processdata"

8-23	7	6	5	4	3	2	1	0
MDC1					A	Q		BDC1

**MDC1** : valeur de mesure du capteur

**A** : bit d'alarme (indique l'existence d'un problème avec la configuration ou la fonctionnalité du capteur)

**Q**: bit de qualité (indique que la qualité du signal est en dessous d'un seuil pré-configuré)

**BDC1**: sortie de commutation logique du capteur

## Données asynchrones

Les données **asynchrones** sont celles qui peuvent être lues ou modifiées sans synchronisation temporelle stricte.

Une partie des paramètres sont montrés ci-dessous à titre d'exemple.

Les **paramètres pré définis** permettent d'identifier le capteur. Les **paramètres des canaux binaires** permettent, par exemple, d'ajuster un seuil à l'intérieur de la plage de détection du capteur.

## IO-Link Pre defined parameters

Index	Subindex (dec)	Access	Parameter name	Coding	Definition
0x000C (12)	0	R/W	Device Access Locks	Uint16	0: Unlocked (default)  1: Device is operating properly
0x0010 (16)	0	R	Vendor Name	String	Baumer Electric AG
0x0011 (17)	0	R	Vendor Text	String	<a href="http://www.baumer.com">www.baumer.com</a>
0x0012 (18)	0	R	Device Name	String	Product Key External ()
0x0013 (19)	0	R	Product Id	String	Baumer Article Number
0x0014 (20)	0	R	Device Text	String	Sensor specific
0x0015 (21)	0	R	Serial Number	String	Production Order Nr / Serial Nr
0x0017 (23)	0	R	Firmware Revision	String	Major.Minor “##.##”
0x0018 (24)	0	R/W	Application Specific Tag	String	Default: Filled with *****, as recommended by the IO-Link spec.
0x0024 (36)	0	R	Device Status	Uint16	0: Device is operating properly

<b>Index</b>	<b>Subindex (dec)</b>	<b>Access</b>	<b>Parameter name</b>	<b>Coding</b>	<b>Definition</b>
					1: Device is operating properly
					2: Out-of-Specification
					3: Functional-Check
					4: Failure
					5 - 255: Reserved
0x0025 (37)	0	R	Detailed Device Status	Uint16	EventQualifier “0x00” EventCode “0x00, 0x00”

## IO-Link Binary Data Channels

<b>Index</b>	<b>Subindex (dec)</b>	<b>Access</b>	<b>Parameter name</b>	<b>Coding</b>	<b>Definition</b>
0x003c (60)	01	R/W	Setpoint SP1	Uint16	Teach Point [mm] (TP)
	02	R/W	Setpoint SP2	Uint16	Not supported
0x003d (61)	01	R/W	Switchpoint logic	Uint8	0x00: not inverted 0x01: inverted
	02	R/(W)	Switchpoint mode	Uint8	Fixed value 0x01: Single point mode

La gestion des paramètres asynchrones peut se faire directement depuis le PLC.

Cependant, la nature du PLC, qui est principalement dédiée à des tâches cycliques, le rend peu efficace pour ce genre de travail.

Une autre possibilité est d'utiliser les logiciels fournis par les fournisseurs qui permettent de

paramétrer les capteurs sans passer par le PLC.

On peut citer, à titre d'exemple, le logiciel de l'entreprise "Baumer" : [Baumer Sensor Suite \(BSS\)](#).

Pour qu'un PLC puisse accéder à un capteur IO-Link, un **IO-Link Master** est nécessaire.

Ce module agit comme une passerelle permettant d'établir une communication bidirectionnelle entre le protocole "IO-Link" et le protocole du bus de terrain du PLC (Profinet, EtherCAT, Ethernet/IP, Modbus TCP, etc.).



IO-Link master Profinet 8 Port IP67

## Travail pratique

### Objectif :

Programmer un bloc fonctionnel (FB) qui lit et traite les données synchrones du capteur de distance en y intégrant certaines fonctionnalités.

# Description de l'interface du bloc fonctionnel (FB)

## Input

Name	Type	Description
Enable	BOOL	Activate Function Block, set data value in output if valid.
HighThreshold	REAL	Sets upper switching threshold
LowThreshold	REAL	Sets lower switching threshold
DefaultOut	REAL	Default value when problems occur

## In Out

Name	Type	Description
hw	UA_O300_DL	In the particular context of the ctrlX to S7 interface.

## Output

Name	Type	Description
InOperation	BOOL	Valid data at output
Value	REAL	Distance from object in mm
HighLimit	BOOL	Distance above HighThreshold
LowLimit	BOOL	Distance below LowThreshold
Error	BOOL	There is an error
ErrorID	WORD	Some details about the error with Error Code.

ErrorID Code	Description
16#0000	Data valid
16#0001	Quality bit, signal is below the configured threshold.
16#0002	Alarm bit, signal an error in sensor, this alarm has priority over ID 16#0001

ErrorID Code	Description
16#0003	Id not defined

## Comportement du bloc fonctionnel (FB)

Il existe quelques modèles de base qui décrivent le comportement des blocs fonctionnels.

On peut citer à titre d'exemple :

- Le modèle "**In Operation Base**" qui fonctionne de manière continue.
- Le modèle "**Execute**" qui fonctionne sur le principe d'un déclenchement ponctuel.

La gestion d'un capteur est un exemple typique du fonctionnement d'un FB selon le modèle "**In Operation Base**".

Le FB est exécuté aussi longtemps que son entrée `Enable` est activée.

Le diagramme d'état d'un bloc fonctionnel de type **In Operation Base** peut se représenter de la manière suivante :

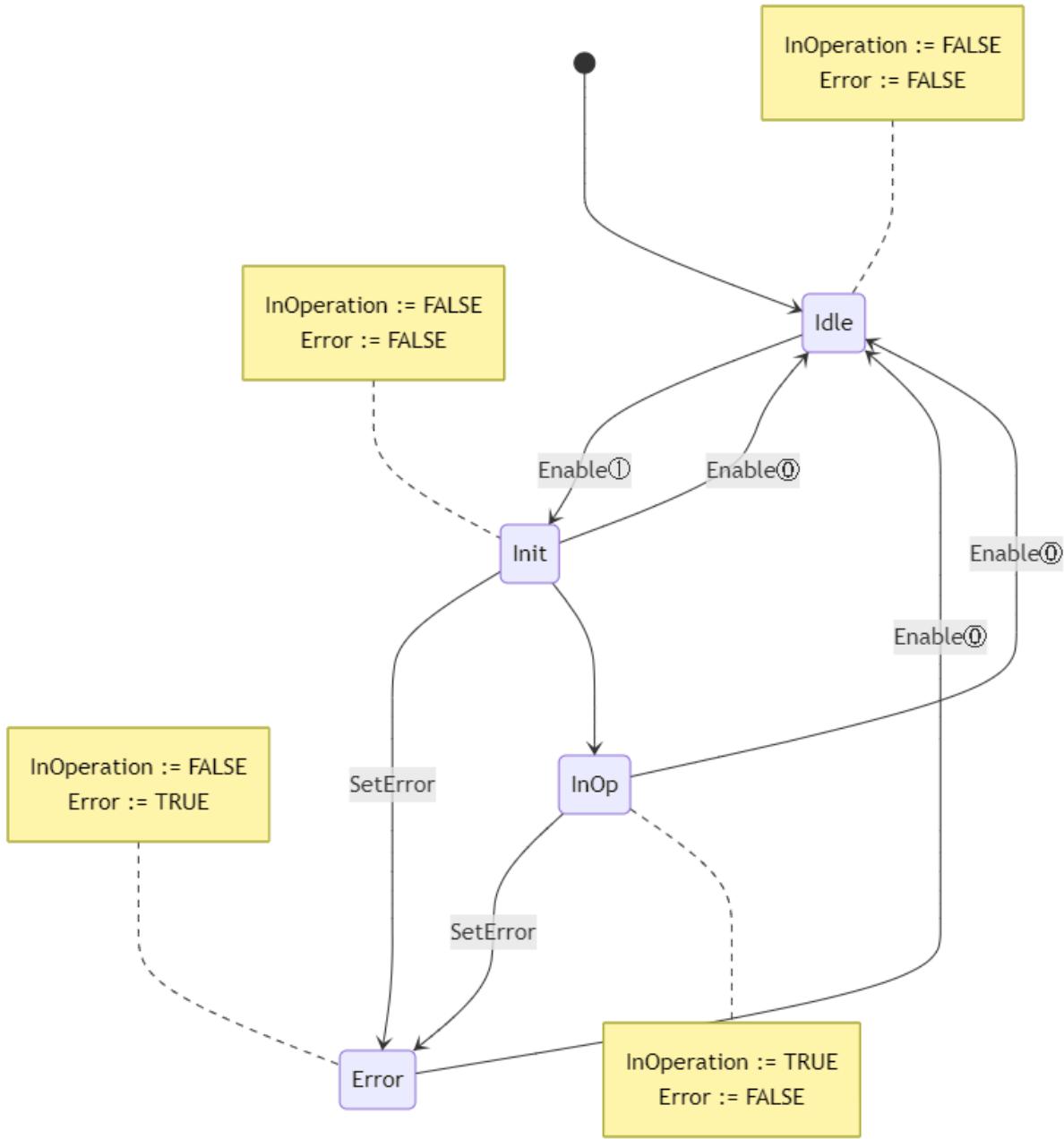


Diagramme état de base

Ce diagramme d'état peut être amélioré en ajoutant un "état globale" InOp intégrant les sous-états suivants :

- Level0k
- LevelLow
- LevelHigh

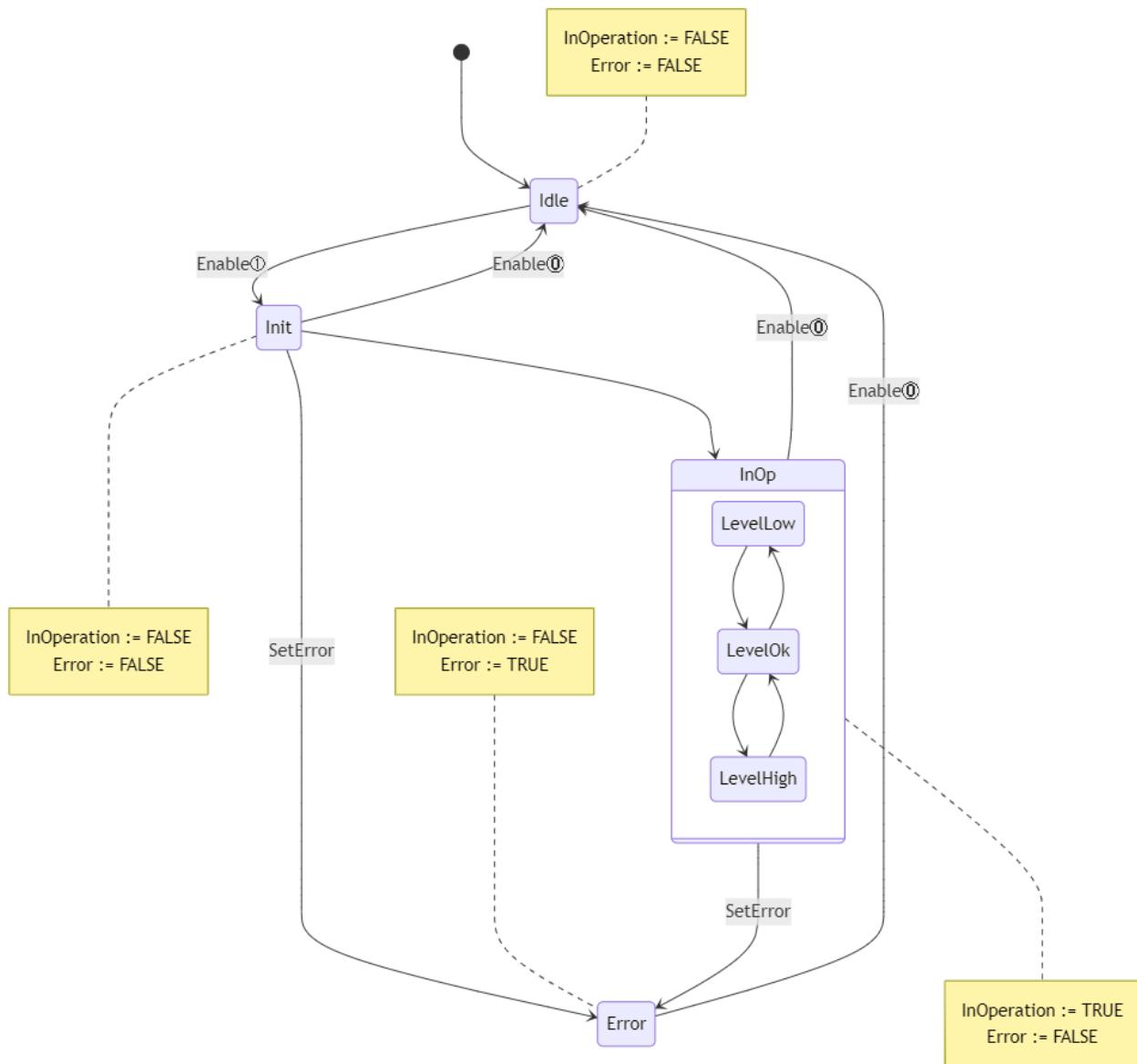


Diagramme d'état avec sous-états

Cette variante peut s'avérer cependant un peu complexe à programmer. Elle sera abordée dans le cadre d'un prochain travail pratique.

N.B.: Il serait théoriquement possible de passer directement de **LevelLow** à **LevelHigh**, mais cela n'apporte rien au fonctionnement général et implique des transitions supplémentaires.

Une autre manière de représenter ce diagramme d'état est d'utiliser tous les états distinctement.

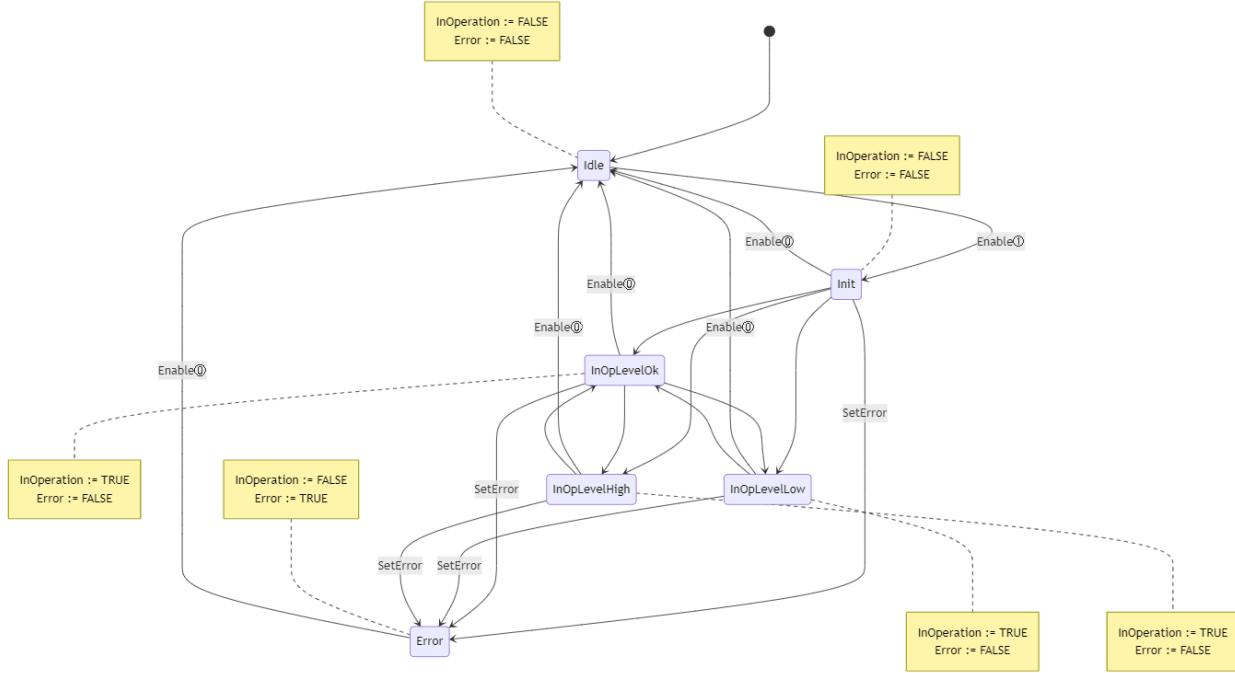


Diagramme état distincts

On constate très vite, que même si le nombre d'états est limité, la représentation complète devient vite difficile à lire.

**Cette représentation sera néanmoins utilisée pour programmer notre bloc fonctionnel (FB).**

## Signal de sortie en cas de dysfonctionnement ou de perturbations

Selon l'utilisation du capteur, une valeur lue erronée peut poser un problème.

Si le capteur est en défaut (Alarm bit signal) ou si la qualité du signal (Quality bit signal) n'est pas correcte, **nous ne voulons surtout pas que signal de sortie value soit à 0.**

Dans ce cas, l'entrée DefaultOut sera affectée à la sortie Value .

Remarque : Le système pourra être testé avec une valeur par défaut de DefaultOut fixée à 251 (soit 1 [mm] de plus que la plage de mesure théorique du capteur).

# Travail à réaliser

1. Programmer le DUT (Data User Type) `E_OperationBaseDL` de type ENUM (énumération) contenant les différents états possibles du capteur.
2. Programmer le FB (Function Block) `FB_0300_DL` en utilisant l'interface ci-dessous.

```
FUNCTION_BLOCK FB_0300_DL
VAR_INPUT
    Enable: BOOL;
    HighThreshold: REAL;
    LowThreshold: REAL;
    DefaultOut: REAL;
END_VAR
VAR_IN_OUT
    hw: UA_0300_DL;
END_VAR
VAR_OUTPUT
    InOperation: BOOL;
    Value: REAL;
    HighLimit: BOOL;
    LowLimit: BOOL;
    Error: BOOL;
    ErrorId: WORD;
END_VAR
VAR
    eOperationBaseDL: E_OperationBaseDL;
    inputConverted: REAL;
END_VAR
```

## Consignes à respecter :

- Ecrire une machine d'état CASE...OF avec les différents états et les conditions de transition.
- Affecter l'état des sorties, `InOperation`, `HighLimit`, `LowLimit` et `Error` en ne tenant compte que des états.
- Définir la variable `ErrorId` avec des conditions IF..ELSIF..ELSE .
- Affecter une valeur à la variable de sortie `Value` à l'aide de IF..ELSE en fonction de l'état de `InOperation` .

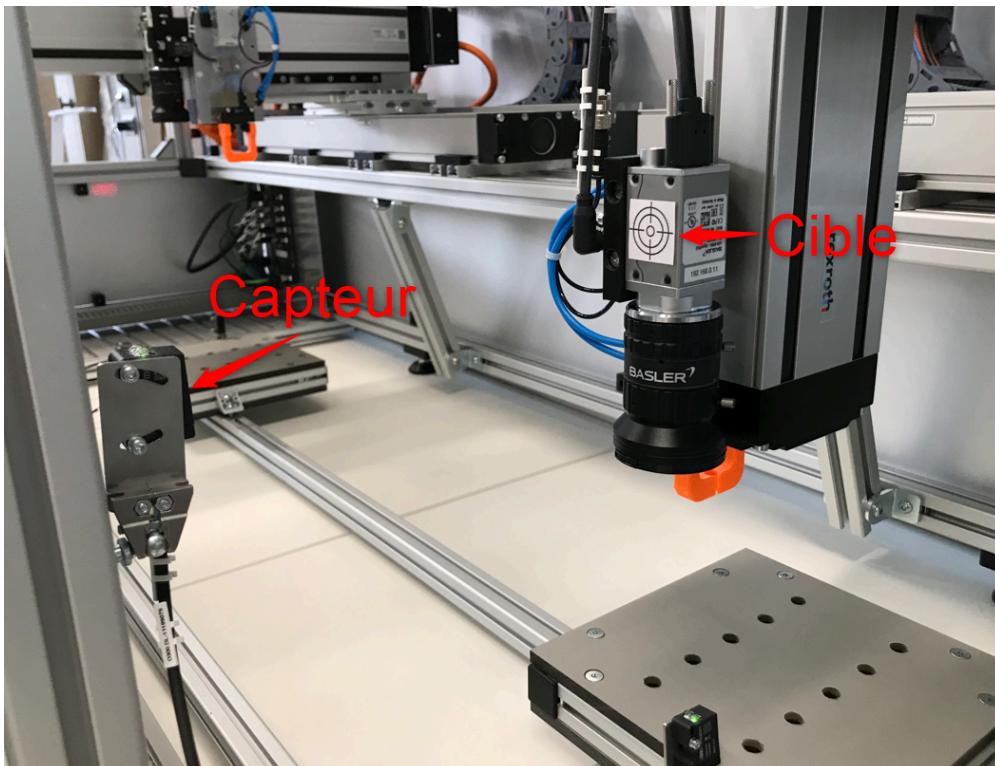
N.B. : Ne pas oublier d'appeler le bloc fonctionnel `FB_0300_DL` depuis le programme principal `PRG_Student` !

#### Information :

Nom de la variable structurée qui permet d'accéder aux données synchrones du capteur de distance :  
`GVL_Abox.uaAboxInterface.ua0300_DL_Optic`

### 3. Tester le fonctionnement du bloc fonctionnel (FB)

La vérification du fonctionnement du FB sera réalisée à l'aide du capteur de distance monté sur l'unité. La position de la cible montée sur le système d'axes cartésiens de l'unité permettra d'ajuster la distance mesurée par le capteur.



Capteur/Cible intégrés dans l'unité

Le déplacement des axes X, Y et Z de l'unité sera réalisé avec l'application **Node-RED** mise à disposition.

#### Marche à suivre pour utiliser l'application "Node-RED" :

1. Remplacer le fichier `flows.json` qui se trouve dans le répertoire `C:\Users\YourUsername\.node-red` par le fichier `flows.json` mis à disposition dans Github.

## 2. Démarrer Node-RED

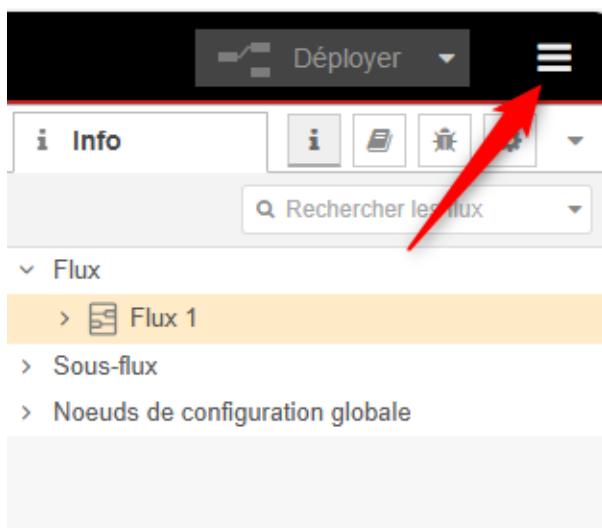
- Ouvrir l'invite de commande (→ cmd.exe)
- Entrer la commande : cd C:\Users\YourUsername\AppData\Roaming\npm
- Entrer la commande : node-red

## 3. Accéder à l'interface utilisateur de Node-RED

- Ouvrir le navigateur
- Entrer l'URL : <http://localhost:1880>

## 4. Installer les modules supplémentaires pour l'application (si nécessaire !)

- Cliquer sur les 3 traits horizontaux en haut à droite



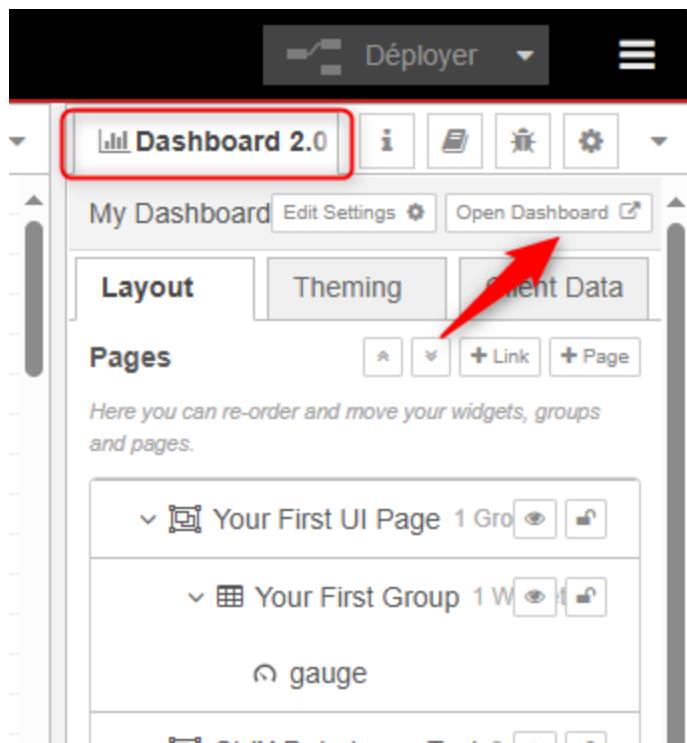
--> Gérer la palette

--> Sélectionner l'onglet "Installer"

--> Installer les modules suivants :

- @flowfuse/node-red-dashboard
- node-red-contrib-ctrlx-automation

## 5. Ouvrir le Dashboard



Ouvrir le Dashboard

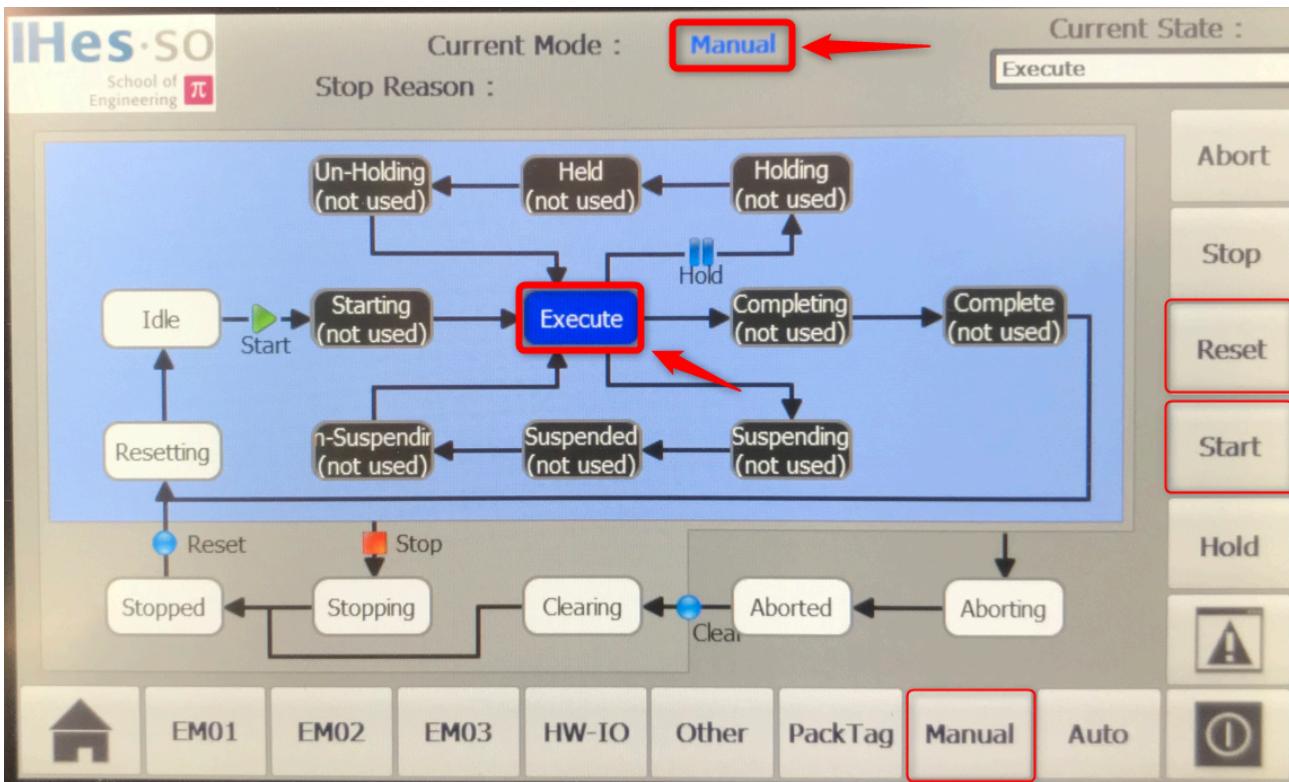
Résultat :



Dashboard de l'application Node-RED

## 6. Encencher l'installation à l'aide du panneau opérateur "Siemens"

- Sélectionner le mode "Manual" à l'aide du bouton `Manual`.
- Mettre l'installation dans l'état "Execute" à l'aide des boutons `Reset` et `Start`.



IHM sur Panneau Opérateur "Siemens"

7. Déplacer la cible en face du capteur de distance l'aide du Dashboard "Node-RED" et effectuer différents tests pour valider le fonctionnement correct du bloc fonctionnel `FB_0300_DL`.
8. Attention, la qualité du signal n'est pas exceptionnelle, en raison de la qualité du papier de la cible. Une surface métallique conviendrait mieux. Il est donc possible qu'il soit nécessaire d'utiliser le logiciel Baumer Sensor Suite, il faut s'enregistrer avec son mail [xxx@hes-so.ch](mailto:xxx@hes-so.ch), pour modifier le seuil du signal Q.

## Warning

Nous utiliserons des alarmes lors du prochain labo. Si une alarme transmet une commande à la machine, ce n'est pas le cas du Warning qui ne fait que transmettre une information **utile** à l'opérateur.

Nous allons utiliser un Warning pour transmettre une information en cas d'erreur du `FB_0300_DL`.

## ID

**ID doit être unique**, car c'est cette valeur que le système utilise pour trier les alarmes. C'est aussi un principe, pour une machine, chaque alarme possède un numéro d'identification unique, même si on peut ajouter un paramètre **Value** et éventuellement modifier une partie du texte. Il serait ainsi

envisageable d'avoir la même alarme pour close et open avec une **Value** et un **Message** différent.  
Mais à condition que **Category** ne change pas.

VAR

```
fbSetWarning_Sensor : FB_HEVS_SetWarning;
```

END\_VAR

## Code

```
fbSetWarning_Sensor(bSetWarning := ...,
                     bAckWarningTrig := FALSE,
                     // Warning Parameters
                     ID := 3,
                     Value := 33,
                     Message := 'Warning 3, Finished',
                     Category := E_EventCategory.Warning,
                     // Reference to plc time from PackTag
                     plcDateTimePack := PackTag.Admin.PLCDDateTime,
                     // Link to PackTag Admin
                     stAdminWarning := PackTag.Admin.Warning);
```

## Conclusion

Les FBs offrent une **abstraction** et une **modularité** qui facilitent la gestion des systèmes complexes, rendent le programme plus lisible et maintenable, et permettent une réutilisation optimisée.