16/04/2025

Puippe Marcelin

HES-SO

TB - 2025

TB – PLCSoft - start

Rapport intermédiaire

# Table des matières

[1. Table des matières 1](#_Toc196569898)

[2. Environnement de développement 2](#_Toc196569899)

[3. Introduction 2](#_Toc196569900)

[4. Point de départ 2](#_Toc196569901)

[5. Objectif 3](#_Toc196569902)

[5.1. Idées d’amélioration et extensions du frontend web 3](#_Toc196569903)

[5.2. L’implémentation bloc de haut niveau 4](#_Toc196569904)

[6. Erreur non géré 5](#_Toc196569905)

[6.1. Manque lien 5](#_Toc196569906)

[6.2. Boucle 6](#_Toc196569907)

[7. Wdx 6](#_Toc196569908)

[7.1. Accéder IO 6](#_Toc196569909)

[Sans wda 6](#_Toc196569910)

[Avec wda 7](#_Toc196569911)

[8. Ressource utile pour la suite 7](#_Toc196569912)

[8.1. n8n 8](#_Toc196569913)

[8.2. total js 10](#_Toc196569914)

[9. Note rapport 11](#_Toc196569915)

[9.1. Transmission information entre UI et MAIN 11](#_Toc196569916)

[Main 🡪 UI 11](#_Toc196569917)

[10. Point d’arrivé (synthèse) 11](#_Toc196569918)

[10.1. Visu 11](#_Toc196569919)

[11. Conclusion 12](#_Toc196569920)

[12. Amélioration possible 13](#_Toc196569921)

[13. Date et signature 13](#_Toc196569922)

# Environnement de développement

|  |  |  |  |
| --- | --- | --- | --- |
| **Appareil** | **Adresse IP** | **Utiliser dans soft** |  |
| Automate WAGO CC100 | 192.168.37.134 | softplc-main |  |
| PC | localhost | Softplcui-main |  |

|  |  |  |
| --- | --- | --- |
| **Logiciel** | **Appareil** | **Commentaires** |
| Goland | PC | Développement logiciel |
| Umlet | PC | Réalisation de schéma basé développement |
| Firefox | PC | Permettant meilleur visualisation de WDx |
| WDx | Automate | Dénomination général pour parler de WDM + WDD + WDA  *WDM = WAGO Device Model (standard utiliser pour les WDD) WDD = WAGO Device Description ( sorte de manifeste décrivant ce que le produit met à disposition)*  *WDA = WAGO Device Access (*Accéder aux paramètres et IO) |
| WDA | Automate | Nouvelle Librairie, interface REST, wago accessible par web en JSON. Permettant la récupération des informations de l’automate et le contrôle des Input/output par requête JSON.  https://192.168.1.126/wda/parameters?page[limit]=20000  https://192.168.1.126/wda/parameter-definitions?page[limit]=20000 |
|  |  |  |

# Introduction

Ce document présente ce qui a été fait et ce qui sera fait pour le projet PLCSoft pour wago. Ce projet consiste en l’ajout de fonctionnalité pour un HAL de développent d’automate sur page web. La programmation ce fait sur une page web de manière graphique et modulaire en reliant les différentes fonctions à disposition entre elles. Le but du projet est d’ajouté de nouveau bloc pouvant être utilisé.

# Point de départ

Le projet se construit sur la base de deux programmes déjà développés, lors du TB 2024 :

* **Softplcui-main**  : Gérant l’interface web côté PC.
* **Softplc-main** : Gérant l’activation des entrées / sorties selon le programme build depuis PLC UI.

Le travail effectué précédemment nous prouve la faisabilité du développement d’un tel HAL.

Les fonctionnalités implémentées dans par ce précédent projet sont :

* Digital Input / output
* Analogue Input / output
* Ton (timer retardé à la montée)

# Objectif

L’objectif est l’amélioration et l’implémentation de nouvelles fonctionnalité.

Les tâches devant être réalisées sont :

* Modifier les programmes actuelle pour utiliser la nouvelle interface REST WDA pour piloté l’automate.
* L’implémentation de nouveaux blocs de haut niveau comme CAN, MQTT, WebServer, client/serveur HTTP et autres bloc. Il faudra trouver une solution pour faire ces tâches par programmation en bloc.
* Amélioration et extensions du frontend web.
* Développement d’un banc de test physique et d’une application de démonstration pour une maison connectée.
* Documentation et tests et rédaction du rapport, poster et présentation.

## Idées d’amélioration et extensions du frontend web

Il y a de nombreuse possibilités d’amélioration pour l’interface utilisateur.

* Interdire les lien qui passent sur bloc, intelligence de connexion.
* Interdire boucle de rétroaction (comme codesys) ou les gérer.
* Ajout de blocs logiques contenant un champ (pour Input déjà un peu fait, non-fonctionnel, mais pas de système de seuil).
* Accordions ajout bar de défilement pour descendre pour voir les élément plus bas
* Save dans un fichier au lieu que dans une variable, également implémenter pour restore.
* Améliorer nomenclature éviter les Output pour analogique et digital et input pour constante. Une idée serait un menu déroulant sur le bloc pour choisir le type.
* Permettre de coder le frontend sans le backend. Donc générer l’accordions à partir d’un fichier qui peut être mis à jour lorsqu’on est connecté.
* Ajout de raccourcis clavier :
  + ctrl + C / V / A / Z / Y.
  + Cliqué glisser = multi sélection. Touche Delete fonctionne.
  + O pour placer output, I pour input.
  + Espace placer un composant du même que le précédent qu’on a placé.
  + Shift + clique gauche glisser pour dupliquer.
  + Une autre idée intéressante serait une touche (ctrl + shift + A) pour automatiquement ajouter tous les blocs autour d’un bloc ou d’un groupe sélectionné avec des valeurs par défauts. Les blocs rajouté seraient les plus proche pour finir le réseau. Par exemple, on place un bloc TON, on le sélectionne, on appuie sur une touche et nous voilà avec une constante de 1 seconde, une input DIO1 (ou la suivante si déjà utilisé) et une output DO1. Pas obligatoire de mettre valeur par défaut.
  + Touche pour activé ou désactivé valeur par défaut
* Ordre input, output, logical … dans l’accordion sont jamais à la même place. Ce qui est pénible car on ne peut pas s’habituer.
* Amélioration du visuel, couleur et autres.
* Ajout d’une barre de menu en haut.
  + Affichage des raccourcis
  + Aide
  + Permettre à l’utilisateur de choisir l’emplacement d’enregistrement.
  + Ajout d’un mode Debug.

## L’implémentation bloc de haut niveau

Il existe plusieurs manière d’aborder le problème. Une des approches est de repérer les points commun entre ces blocs de haut niveau pour essayer d’en tirer une forme générique. On remarque que tous ces blocs ont pour objectif de transmettre et recevoir des données. Il faudra donc commencer par le développement de bloc commun pour une communication. Il faut également des blocs permettant de travailler avec des STRING. Le schéma *figure 1* montre le concept d’une telle structure avec tous les blocs qui devront être développé autour pour pouvoir créer une communication.

L’idée étant d’avoir un bloc qui s’occupe de la configuration étant différente pour can, Mqtt etc. Un bloc de communication qui permet de lier tous les blocs et par le future permettra à l’utilisateur de voir l’état de la communication (en mode debug).

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure : Communication objectif

# Erreur non géré

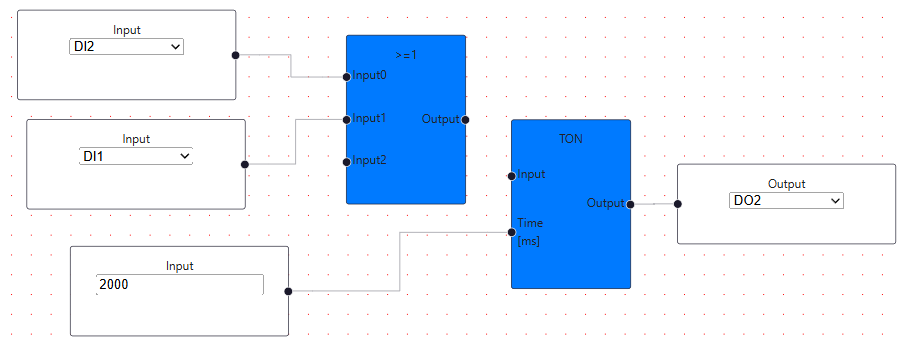
Certaines erreurs n’ ont pas été traitées lors de l’ancien TB.

## Manque lien

Le problème est que le programme plcSoft plante au lieu d’afficher simplement une erreur et de ne pas Build le programme dans l’automate.

Cependant le save est possible et le restore peut être fait après avoir relancé le programme.

**Remarque** Bloc bleu : il y a des bloc bleu qui est le résultat d’un test explication dans la synthèse.



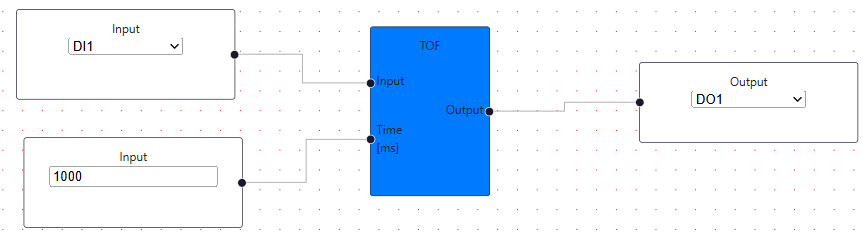


Figure : Défaut manque lien entre OR et TON

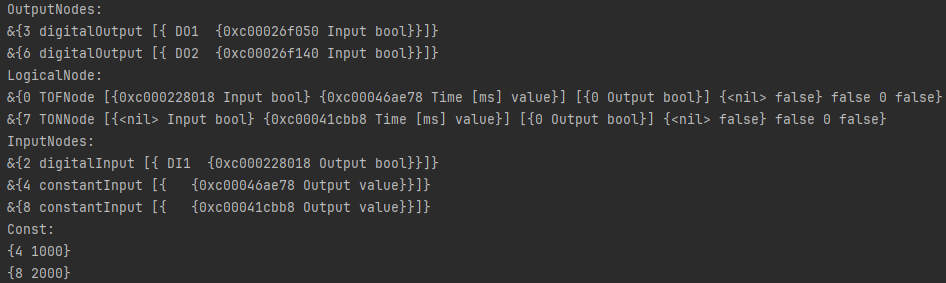


Figure : message plcsoft save défaut

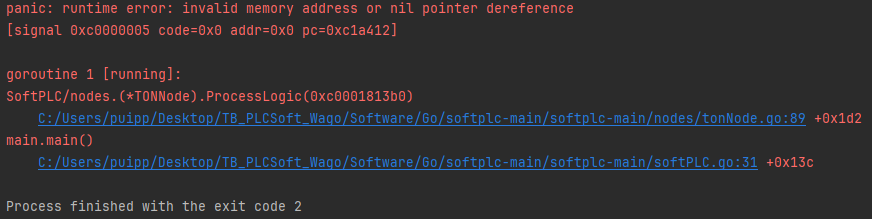
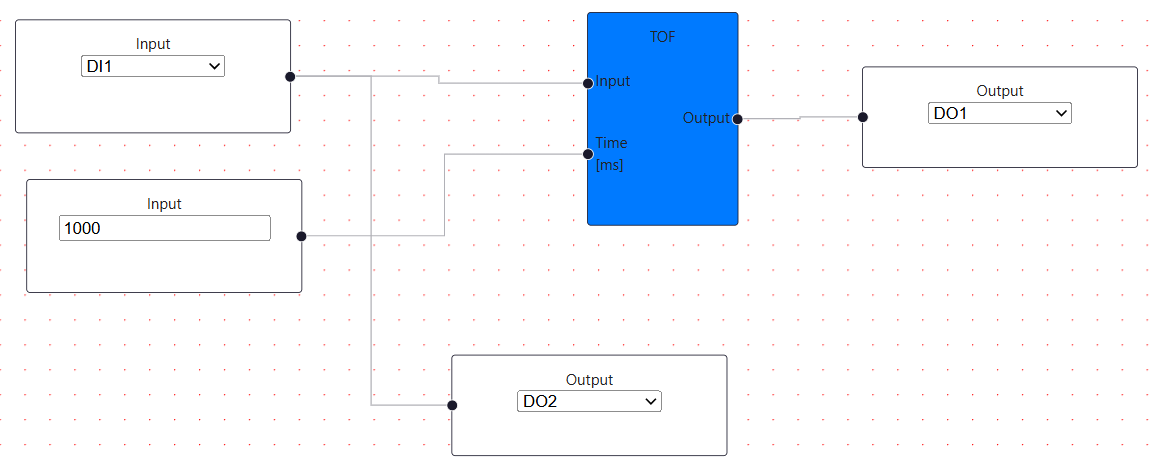


Figure : Erreur Build manque lien entre OR et TON

## Boucle



Doit être possible, mais est interdit.

# Wdx

Documentation API : <https://192.168.37.134/openapi/wda.openapi.html>  
Documentation : <https://downloadcenter.wago.com/wago/null/details/m2ddbfyuihrn44rp2h>  
json Parameter : [https://192.168.37.134/wda/parameter-definitions?page[limit]=20000](https://192.168.37.134/wda/parameter-definitions?page%5blimit%5d=20000)

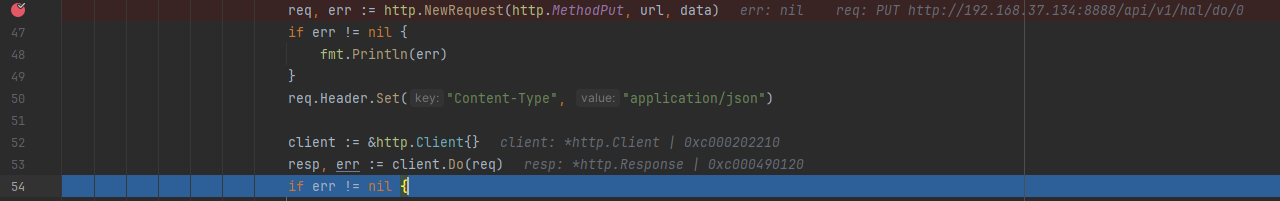
## Accéder IO

### Sans wda

[192.168.37.134:8888/api/v1/hal/io](http://192.168.37.134:8888/api/v1/hal/io)  
{"di":[false,false,false,false,false,false,false,false],"do":[false,false,false,false],"ai":[0.336,0.343],"ao":[0,0],"temp":[16778.26508951407,16778.26508951407]}

#### Activer output

Dans fichier « OutputUpdate.go » de softplc-main.

Pour active DO1 : PUT http://192.168.37.134:8888/api/v1/hal/do/0  


C’est à partir de la ligne 54 qu’on a la lampe allumé. **(mtn 73)**

Plus de détaille dans dossier « autre » puis dossier « Request ».

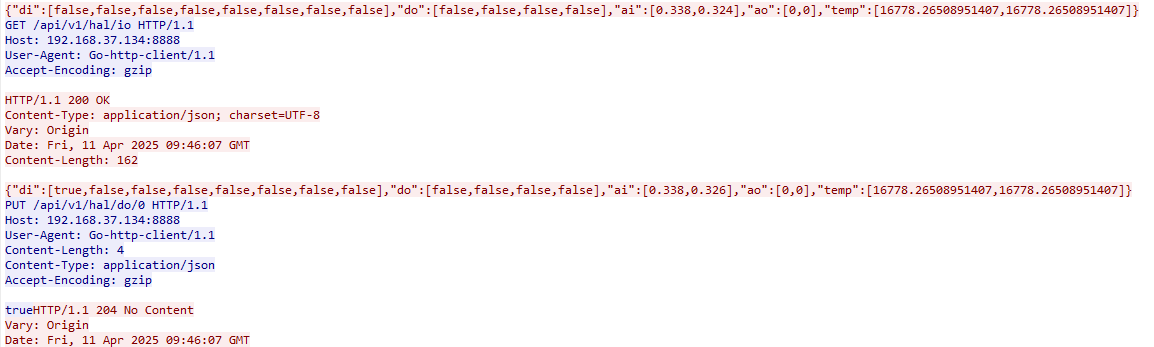
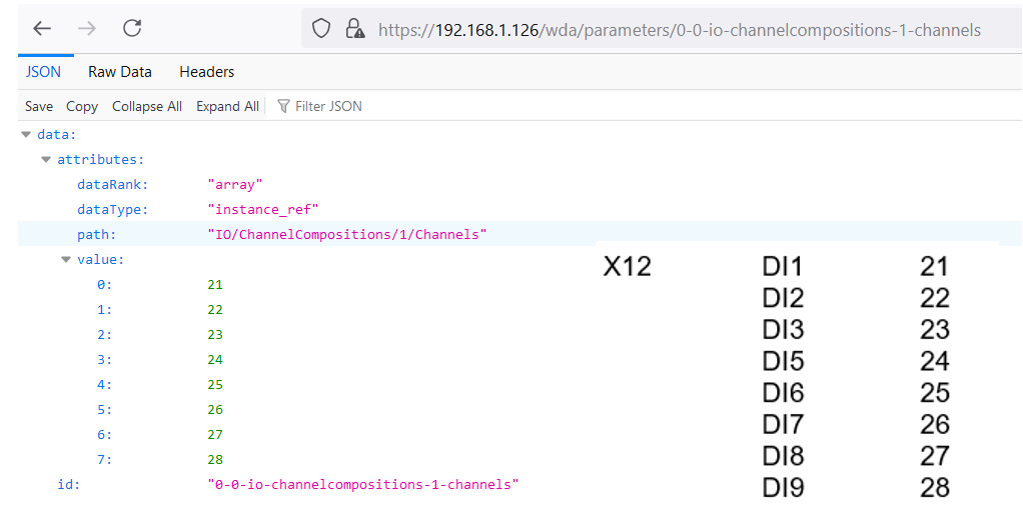


Figure 5 : wireShark Http stream

### Avec wda

751-9402 : <https://192.168.37.134/wda/parameters/0-0-io-channelcompositions-1-channels>



C’est possible avec un automate 751-9402, cependant on a le modèle 751-9401.

# Ressource utile pour la suite

Durant le cours projet 4, plusieurs ressource pouvant être utile pour la suite ont été trouvé.

## n8n

C’est un logiciel d’automation présent en ligne sur gitHub. Il permet une programmation en no-code sur page web comme ce qu’on l’on essaie de faire. Il est surtout conçu pour l’automation de tâche simple. Il permet notamment l’automation de chat alimenté par l’IA, c’est-à-dire des réponses automatiques. Ce n’est pas ce qui nous intéresse mais cela peut nous aider à avoir des idées.

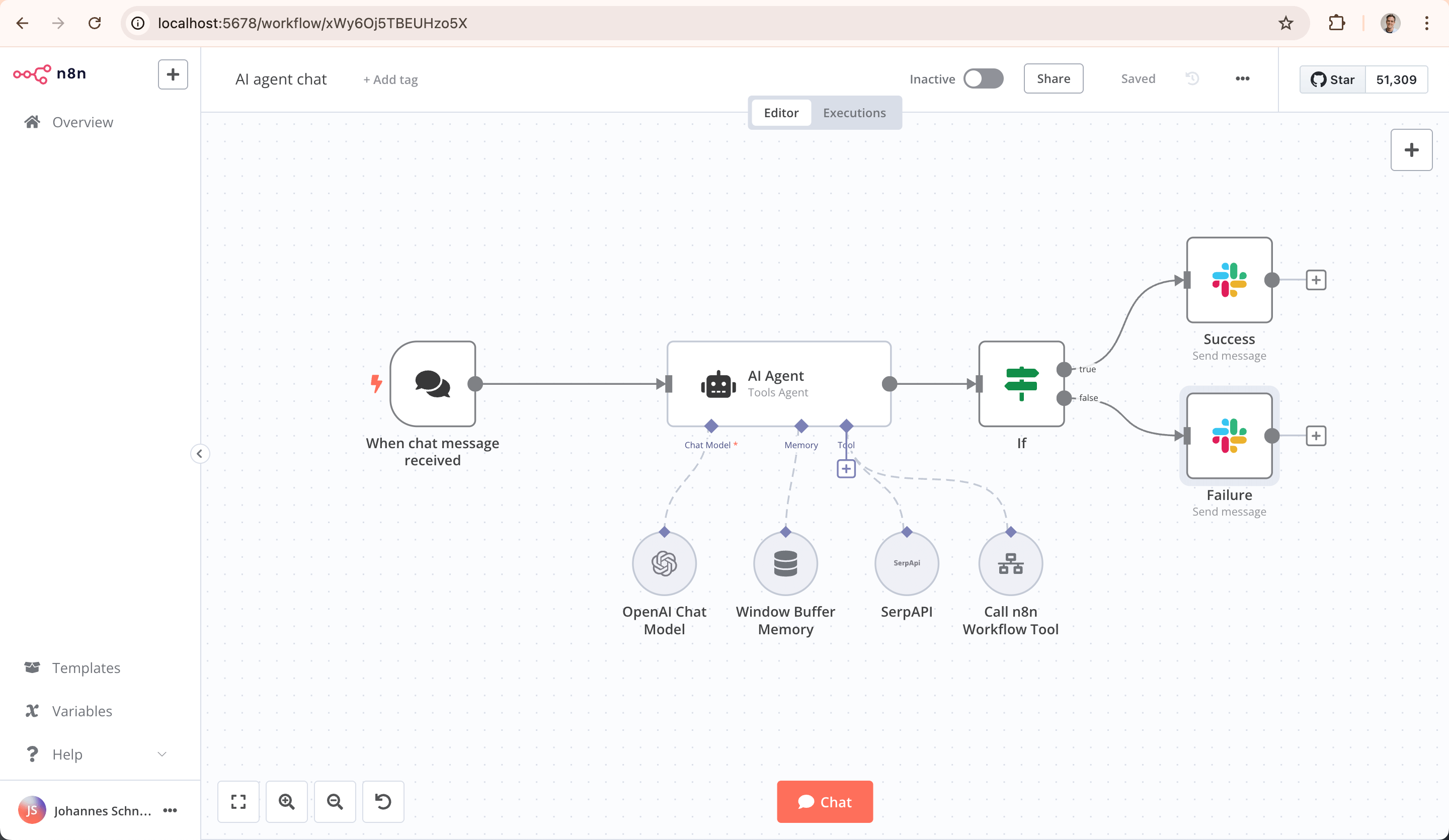


Figure : Exemple gitHub n8n

L’activation d’une output en n8n peut ce faire de la manière suivante. Il suffit d’un bloc HTTP Request1 qu’on configure.

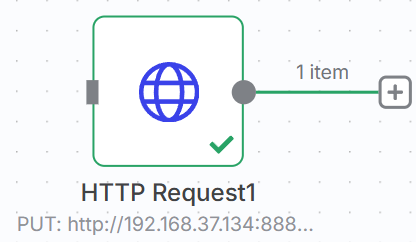


Figure : Bloc HTTP Request1 en n8n

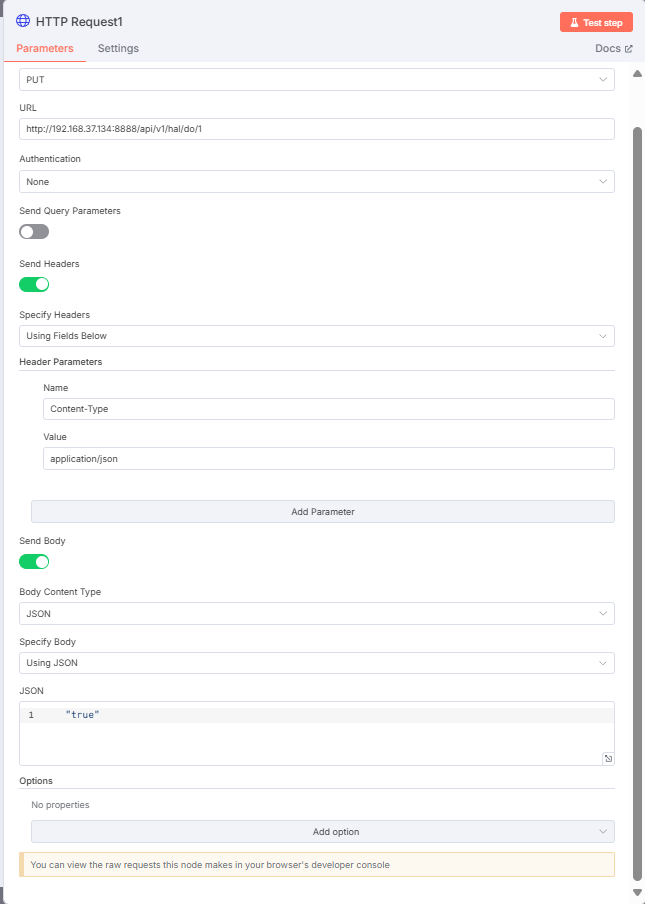


Figure : configuration HTTP Request1 en n8n pour activation output 2

## total js

# Note rapport

## Transmission information entre UI et MAIN

### Main 🡪 UI

***BlockDescriptor*** : file JSON à l’initialisation

Possibilité pour mode **debug**, au moment d’un *updateOutput*:

* Récupérer les états des inputs digital : *\*nodeOutputList.OutputHandle.Input*

# Point d’arrivé (synthèse)

L’automate a été câblé et configuré. Il est prêt à être utilisé pour la suite du projet.

Les programmes softplc-main et softplcui-main ont pu être testés et fonctionne comme décrit dans le travaille de TB précédent. Cependant, l’**analogique** n’a pas été testé mais ne fonctionne sûrement pas car ne figure pas dans les points qui ont été validés tout comme **appliance Input** qui ne fonctionne pas et fait planté l’interface. De plus, de nombreuse amélioration décrite dans la partie « objectif » sont possibles.

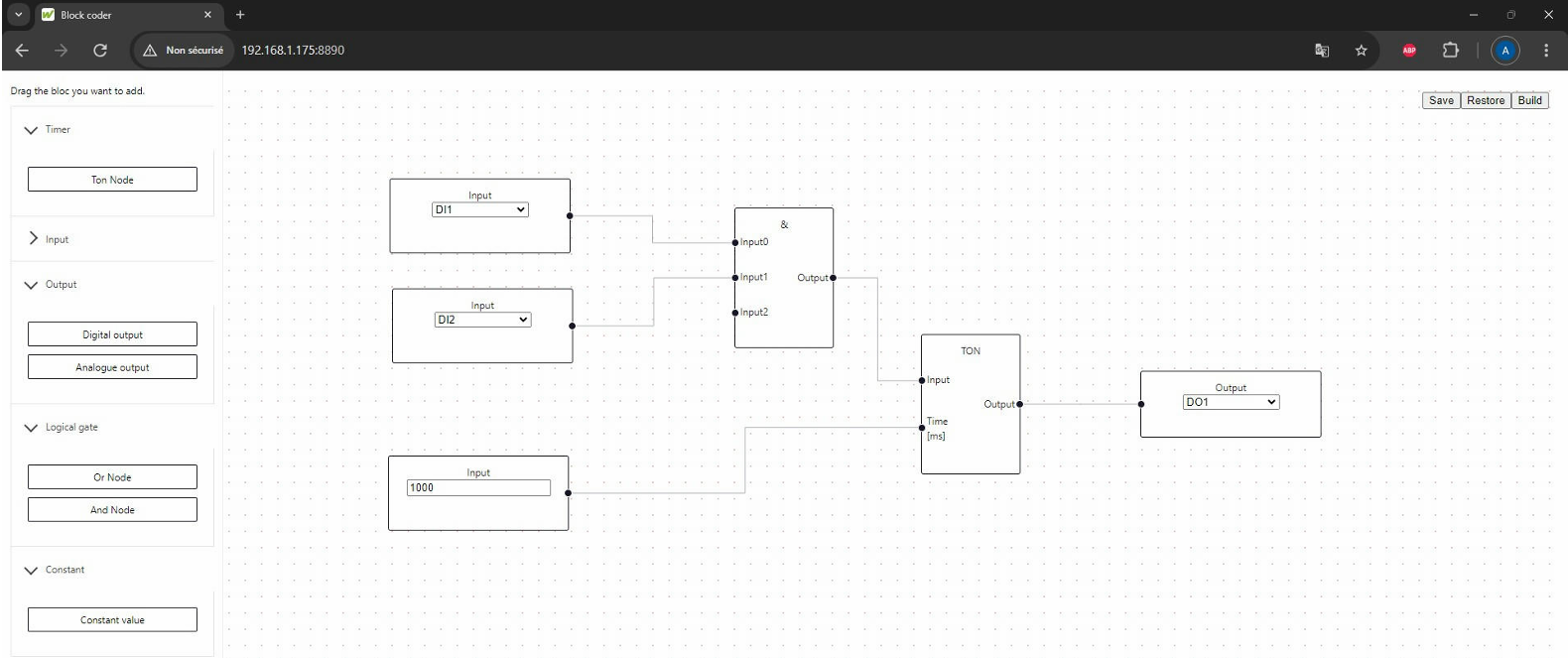
Fonctionnalités développées :

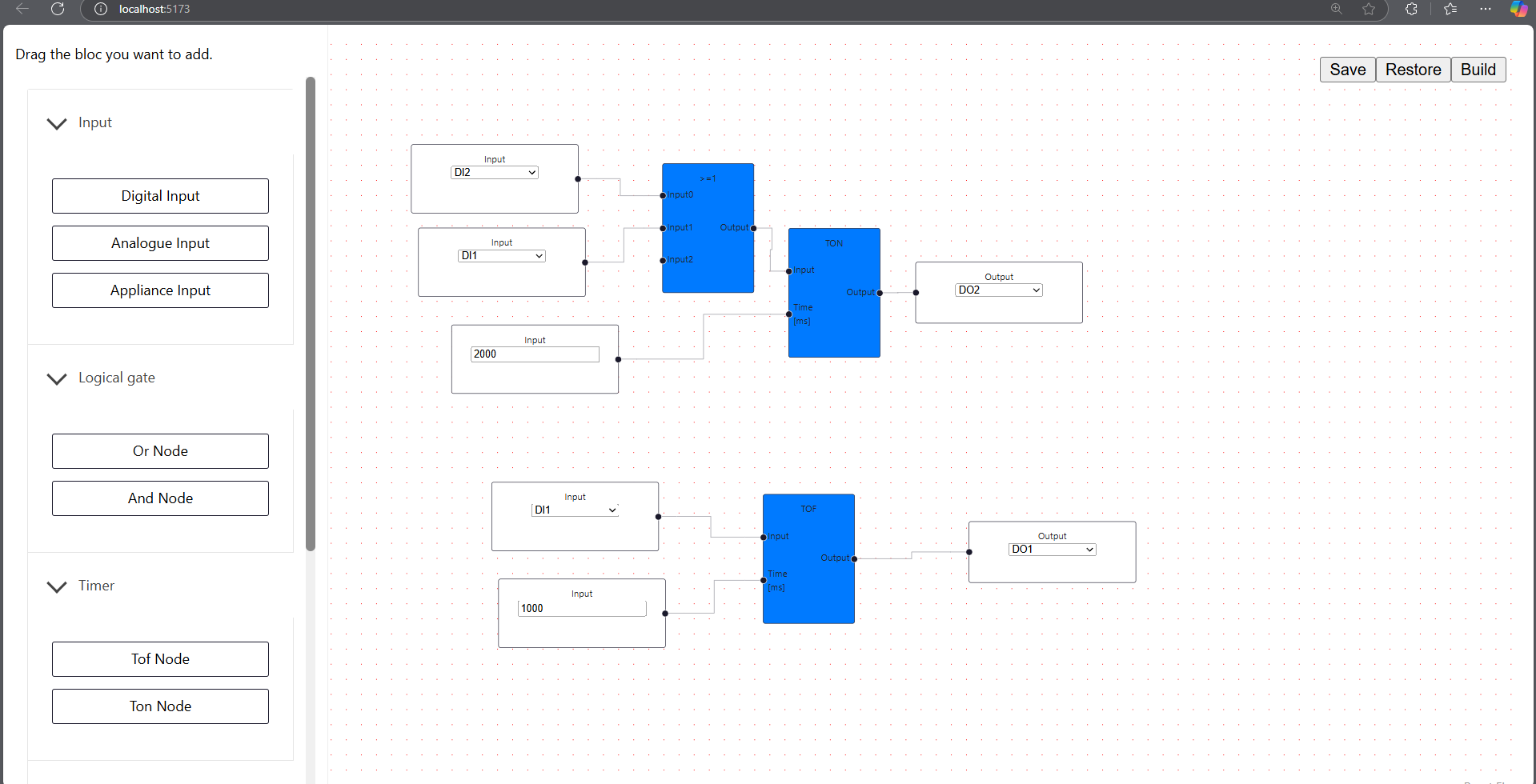
* Un nouveau bloc timer de type TOR a été ajouté. Prouve la faisabilité de rajout de bloc simple. Montre que le programme est robuste et est adaptatif.
* Ajout de la fonctionnalité qui enregistre dans un fichier et restore à partir de celui-ci. Très pratique, elle nous évite de devoir réécrire le code tous le temps à chaque changement et chargement du programme.
* Ajout d’une SlideBar. C’était nécessaire.

Il a été décidé de surtout rajouté des fonctionnalité pour faire gagner du temps.

## Visu

**Avant** :



**Après** : 

La différence est la slide Bar car avant si on ouvrait tous on n’avait pas accès aux composants du bas.

**Remarque** Bloc bleu : il y a des bloc bleu qui est le résultat d’un test qui a été fait l’objectif était de voir comment était gérer le style css des blocs. Le résultat est qu’il est géré par groupe. Ainsi, tous les Blocs LogicalNode ont le même type. Il faudra donc améliorer la structure pour rendre plus facile l’attribution de style.

# Conclusion

# Amélioration possible

Mode debug : web socket   
Mqtt inscription au topic

# Date et signature

Sion, le 19.12.2024

Marcelin Puippe

Une image contenant écriture manuscrite, Police, calligraphie, typographie

Description générée automatiquement

Bibliographie