Rapport technique module 3

**24.06.2025 | EST Porrentruy**

M1 Communication

M3 Informatique industrielle

Candidat : B. Hofer & L.Domon

Responsable de filière : M. C.Cailler

Responsable du module 1 : M. W.Grüter

Responsable du module 3 : M.D. Montavon

Table des matières

[Préface 3](#_Toc188485186)

[Introduction 3](#_Toc188485187)

[Description du projet 3](#_Toc188485188)

[Planification 3](#_Toc188485189)

[Travaux rendus 3](#_Toc188485190)

[1. Description 4](#_Toc188485191)

[2. Mode et fonctionnement 4](#_Toc188485192)

[2.1 Fonctionnement global 4](#_Toc188485193)

[2.2 Mode manuel 5](#_Toc188485194)

[2.3 Mode initialisation 5](#_Toc188485195)

[2.4 Mode automatique 5](#_Toc188485196)

[2.4.1 Mode automatique simple 6](#_Toc188485197)

[2.4.2 Mode automatique pas à pas 6](#_Toc188485198)

[2.4.3 Mode paramétrable 7](#_Toc188485199)

[2.4.4 Mode recette 8](#_Toc188485200)

[2.5 Visualisation 9](#_Toc188485201)

[3. Problèmes rencontrés 10](#_Toc188485202)

[3.1 Boutons de la visualisation 10](#_Toc188485203)

[3.2 Arrêt immédiat 10](#_Toc188485204)

[3.3 Réglage de l’heure 10](#_Toc188485205)

[3.4 Autres difficultés 10](#_Toc188485206)

[4. Suivi 11](#_Toc188485207)

[5. Bilan 11](#_Toc188485208)

[Conclusion 11](#_Toc188485209)

[Source et annexes 12](#_Toc188485210)

[Source 1 12](#_Toc188485211)

[Source 2 12](#_Toc188485212)

[Annexe 1 12](#_Toc188485213)

[Annexe 2 12](#_Toc188485214)

[Annexe 3 12](#_Toc188485215)

[Annexe 4 12](#_Toc188485216)

# Préface

Je m’appelle Lucas Domon, j’ai 21 ans et j’habite à Alle. J’ai commencé ma formation professionnelle par un apprentissage de polymécanicien en août 2019 dans l’entreprise Sphinx Tools à Porrentruy. Pour donner suite à ça j’ai continué à travailler dans cette entreprise jusqu’à mon service militaire obligatoire. Je suis actuellement en première année d’école supérieur dans la filière système industriel à plein temps.

# Introduction

Description du projet  
Le rapport que nous devons réaliser fait partie du module 1 et est lié au module 2, car le projet à concevoir porte sur la programmation d’un automate. Ce projet du module 2 consiste à programmer un automate Wago afin de faire fonctionner la station mise à notre disposition. Cette station est principalement constituée d’un moteur, de deux vérins pneumatiques, d’une balise lumineuse, de nombreux capteurs et d’une commande qui contient plusieurs boutons de fonctionnement. Il nous a été demandé de réaliser plusieurs modes, tels que le mode manuel, le mode d’initialisation et plusieurs modes automatiques.

Planification  
Lors de la séance client du 28 octobre 2024, nous avons planifié, avec M. François-Xavier Petignat, les différents jalons à atteindre :

04.11.2024 : début du projet

18.11.2024 : visualisation et mode manuel en simulation

02.12.2024 : mode automatique simple, pas à pas et paramétrable en simulation

13.01.2025 : présentation du module 2

Ces jalons m’ont permis de maintenir un rythme régulier dans l’avancée de mon projet. Ils ont également servi de repères, après chaque correction, pour apporter des modifications au programme.

Travaux rendus  
Nous avons dû rendre plusieurs dossiers en lien avec le module 2. Parmi ceux-ci, il y avait le mandat à soumettre le 2 décembre 2024. Parallèlement, il était nécessaire de remplir un journal de travail détaillé et de rédiger un rapport technique, tous deux liés au module 1.

# Description

Pour concevoir un projet sous codesys, il est nécessaire de créer des POUs, une visualisation, une liste de variables globales, et d’autres éléments. Je vais expliquer mes différents POUs en détaillant la manière dont je les ai réalisés ainsi que les raisons pour lesquelles j’ai fait ces choix. Ils sont classés dans l’ordre alphabétique dans mon programme. Certains de ces POUs doivent être programmés dans un langage spécifique selon les exigences de la séance client.

Programmé en SFC, le POU Cycle comprend mon programme automatique simple, mon mode automatique paramétrable et mon mode automatique recette. Il est structuré en macros pour garantir la clarté et la propreté du code.

L’étape ModeAutomatique correspond à mon mode automatique simple. Le programme fonctionne ainsi : le moteur tourne dans le sens horaire jusqu’à atteindre la position de 6 heures. Ensuite, il effectue un cycle des vérins que j’ai défini. Une fois le cycle terminé, le moteur revient à la position de « 12 heure » et continue son fonctionnement en boucle, jusqu’à ce qu’une demande d’arrêt soit effectuée.

Le POU Cycle représente la majorité de ma programmation. Il permet de gérer le mode automatique simple et pas à pas, le mode paramétrable, ainsi que le mode recette. Mon premier POU doit être écrit en SFC. Le mode manuel, quant à lui, doit être réalisé directement dans la gestion des sorties, en utilisant le langage CFC. Pour le reste du programme, le choix du langage est libre.

# Mode et fonctionnement

## 2.1 Fonctionnement global

Pour bien commencer ce projet, il faut déjà mettre en place de bonnes bases afin de permettre un bon déroulement du travail à rendre. Il m’a été difficile de me structurer, car j’ai très peu de connaissances en programmation. J’ai commencé par faire ce qui me semblait le mieux et le plus logique. Ensuite, je me suis demandé : comment vais-je pouvoir faire la liaison dans mon programme entre les boutons, actionneurs et capteurs physiques et ceux créés dans ma visualisation ?

J’ai commencé par la gestion des sélecteurs (GestionSelecteurs), en faisant la logique en ladder, car c’est plus instinctif pour moi. J’ai pensé ma logique en me disant : le sélecteur physique ou le sélecteur de ma visualisation est égal à une variable globale représentant ce même sélecteur. J’ai rencontré une petite difficulté avec les sélecteurs AUTO et V2, car ils ne s’actionnent que lorsque les sélecteurs de gauche et de droite ne sont pas activés.

J’ai ensuite créé un POU nommé GestionEntrées, qui cette fois a été réalisé en structuré. Ce POU m’a servi à établir le lien entre les différentes entrées. Une fois cette partie mise en place, mon premier problème concernait la gestion du bouton stop et celui d’arrêt d’urgence. Ces boutons sont normalement fermés (NC, "Normally Closed" en anglais), ce qui a été compliqué pour moi, étant davantage habitué à un métier basé sur la mécanique plutôt que sur la logique.

J’ai créé par la suite un POU pour gérer les positions du moteur. De cette manière, je pouvais indiquer que si le moteur franchissait un capteur dans un certain sens de rotation, alors il se trouvait entre deux capteurs spécifiques.

Par exemple : si le moteur tourne dans le sens horaire et qu’il vient de passer le capteur « 6 heures », alors le moteur se trouve entre « 6 heure » et « 9 heure ». J’ai donc noté qu’il était en position « 7 heure ».

## 2.2 Mode manuel

Pour le mode manuel, nous devions le programmer directement dans la gestion des sorties (GestionSorties). Il était demandé de le réaliser en CFC. Plusieurs conditions devaient être mises en place pour permettre le mouvement du moteur ou des vérins. Dans ce POU, nous gérons également la balise lumineuse. Je n’ai pas rencontré de difficulté particulière pour cette partie, car je trouve que le CFC ressemble beaucoup au Ladder.

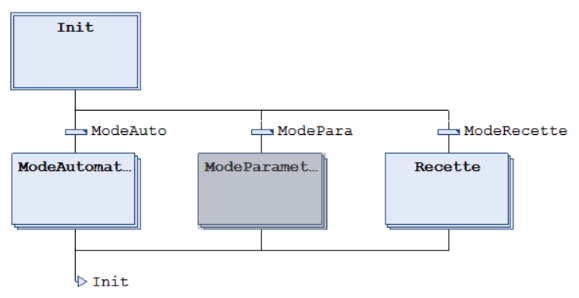
## 2.3 Mode initialisation

Pour faciliter la programmation des modes automatiques, j’ai paramétré un mode d’initialisation. Ce mode consiste, si la machine n’est pas en position initiale, à la remettre en position dès l’appui sur le bouton physique ou celui de la visualisation INIT.

La position initiale correspond au moteur en position « 12 heure » et aux deux vérins en position de repos. Dans mon projet, j’ai donc initialisé la position en programmant que si la station n’est pas en position initiale et que j’appuie sur le bouton quittance, la station se rend automatiquement à la position demandée.

J’ai rencontré une petite difficulté : il était demandé que le moteur utilise le chemin le plus court pour atteindre sa position initiale. Pour cela, j’ai utilisé les positions intermédiaires du moteur afin de définir son sens de rotation pendant l’initialisation.

## 2.4 Mode automatique

Les différents modes automatiques devaient être programmés dans un seul SFC. Ce langage est plus adapté pour créer des programmes répétitifs et cycliques. J’ai commencé par le mode automatique simple.

Cependant, au début, la notion de cycle n’était pas très claire pour moi. Je ne comprenais pas comment faire en sorte que le programme répète le cycle plusieurs fois.

Figure  : Gestion des cycles automatique

### 2.4.1 Mode automatique simple

Figure : Mode automatique simple

Le mode automatique est constitué d’une suite d’étapes et de transitions afin de permettre le déroulement du cycle. On demande d’abord une condition pour entrer dans la macro, puis une condition pour pouvoir lancer le cycle. Par exemple : être en position initiale et appuyer sur le bouton start.

Ensuite, le moteur tourne jusqu’à la position « 6 heure » et effectue un cycle avec les vérins. Mon exemple :

Sortir le vérin 2,

Sortir le vérin 1,

Rentrer le vérin 1,

Et pour finir, rentrer le vérin 2.

Le moteur retourne ensuite à la position « 12 heure » et continue le cycle tant qu’il n’y a pas de demande d’arrêt.

### 2.4.2 Mode automatique pas à pas

Pour le mode pas à pas, j’avais commencé par créer un cycle spécifique, mais après réflexion, j’ai préféré utiliser une variable pour avoir exactement le même cycle et éviter de le recopier.

Description de la variable utilisée : lorsque j’appuie sur le bouton quittance et que le sélecteur mode pas à pas est activé, la valeur en sortie de cette variable correspond à mon GVL.xMarchePP. Cette valeur est en front montant, ce qui signifie que je ne peux pas maintenir le bouton quittance pour continuer mon cycle. Chaque appui doit donc être distinct pour passer à l’étape suivante.

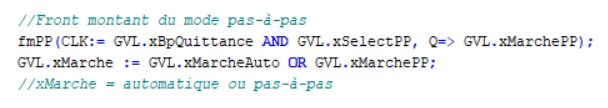


Figure : Configuration du mode pas à pas

### 2.4.3 Mode paramétrable

Le but du mode paramétrable est de pouvoir modeler un cycle plutôt simple et de choisir où le moteur doit s’arrêter et si, après, il doit faire un cycle des vérins ou effectuer un temps d’attente. Pour ma structure de programme, j’ai positionné mes différentes positions d’arrêt et j’ai ensuite fait un branchement pour qu’il effectue soit le cycle de vérins, soit le temps d’attente. Bien sûr, si je ne veux pas qu’il s’arrête à une certaine position, il suffit de ne pas la sélectionner.

Le plus dur dans cette partie du projet était de trouver comment structurer correctement le programme SFC pour qu’on ait le choix de ne pas s’arrêter à une position choisie. Au début, je n’avais pas choisi le type UINT pour les variables de temps, mais après avoir effectué quelques recherches et obtenu des explications, j’ai mieux compris, ce qui m’a facilité la tâche pour le réglage du temps.



Figure : Visualisation du mode paramétrable

### 2.4.4 Mode recette

Le mode recette est loin d’être le mode le plus facile. Il fallait déjà bien comprendre le fonctionnement des tableaux, car dans le cas de l’éditeur de recettes, il faut combiner plusieurs tableaux. L’incrémentation des tableaux a été vue en partie à l’école avant de commencer le mode recette. Cela m’a permis de comprendre une partie de ce mode. Trois tableaux ont dû être créés pour le mode recette : un tableau pour éditer la recette, un tableau pour la recette et un tableau regroupant les recettes stockées (5 recettes différentes).

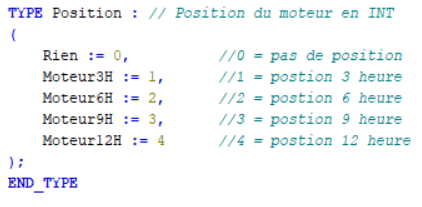
Le premier problème rencontré a été que je ne comprenais pas comment faire pour pouvoir sélectionner une position du moteur, par exemple « 9 heure », et la transformer en ENUM pour pouvoir l’insérer dans mon tableau. Un camarade de deuxième année m’a expliqué comment incrémenter cela avec le morceau de code ci-contre.

Figure 5 : Incrémentation des positions moteurs

Ensuite, une fois que mon SFC était fait et que mon tableau éditeur de recette avait été inséré, j’ai rencontré un problème : mon mode recette se lançait mais ne fonctionnait pas du tout. C’était bizarre, car codesys arrivait à le compiler. J’ai cherché pendant un bon moment avant de me rendre compte que je ne disais pas à ma recette que j’éditais de se mettre dans le tableau de production (ma recette de production était vide). J’ai alors ajouté un bouton pour pouvoir envoyer en production la recette que j’éditais.

J’ai ensuite fait de nombreuses modifications dans ma gestion du cycle pour me permettre, par exemple, de compter correctement la ligne de la recette et passer à la ligne suivante, que ma demande d’arrêt fonctionne correctement, et que le nombre de tours soit correct. Toutes ces modifications m’ont pris beaucoup de temps, plus que je ne le pensais.

## 2.5 Visualisation

Ma visualisation de base apparaît directement à l’accueil et ressemble à l’image ci-dessous. On peut y voir, depuis la gauche, les boutons gris qui servent à basculer sur les autres visualisations que j’ai créées. Ensuite, les boutons blancs sont la représentation des boutons de la commande. Plus au centre, il y a les deux vérins, puis le moteur avec le capteur de distance. Et tout à droite, la balise lumineuse, avec en haut et en bas des indicateurs qui montrent si l’homme mort est actif, si la barrière est activée, si le bouton d’arrêt d’urgence est enclenché, ou si la pression d’air est suffisante. Tout en bas, on peut retrouver les sélecteurs de la commande. Et tout en haut à droite, affichés en rouge, les messages d’erreur ainsi que la date.

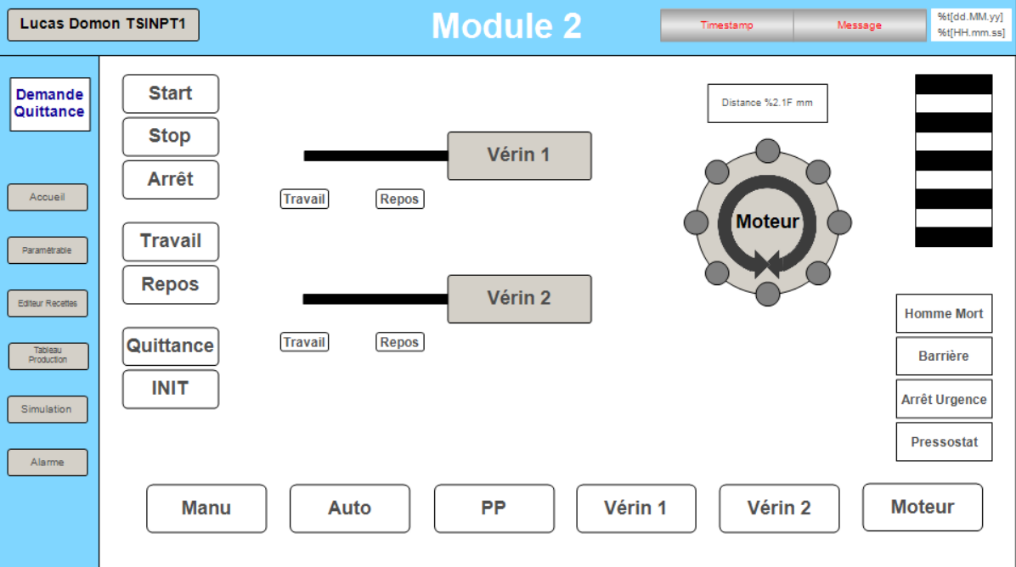


Figure 6 : Accueil de ma visualisation

# Problèmes rencontrés

Je viens d’expliquer mes différents modes que j’ai réalisés, en détaillant les quelques difficultés et problèmes que j’ai rencontrés. Pour ce prochain paragraphe, je vais citer les problèmes concernant le projet en général, des soucis plus globaux que j’ai rencontrés. Parmi ceux-ci, il y a mon arrêt immédiat, le réglage de l’heure de l’automate, certains boutons de la visualisation, et d’autres.

3.1 Boutons de la visualisation  
Pour certains boutons de la visualisation, c’est un camarade automaticien qui nous a montré la fonction **(**OnMouseClick**)**, qui permet d’insérer une valeur, de temps, de comptage ou autre, dans un bouton. Cette fonction m’a beaucoup apporté pour les boutons du mode paramétrable et du mode recette. Elle me permettait d’insérer le temps que je souhaitais ou encore de basculer d’une page de visualisation à une autre.

3.2 Arrêt immédiat  
Mon arrêt immédiat ou double stop fonctionnait, mais ne répondait pas aux critères demandés. Je devais à chaque fois faire une initialisation (remettre la station en position initiale), et je ne pouvais pas redémarrer après un arrêt immédiat. Je suis actuellement en train de résoudre ce problème grâce à SFCPause. Il me reste encore quelques modifications à effectuer pour présenter le projet une deuxième fois.

3.3 Réglage de l’heure  
Pour le réglage de l’heure, je n’ai pas réussi à le faire, malgré mes recherches et tentatives d’essai. Ce n’est que quelques jours après ma présentation qu’un camarade informaticien, en modifiant le code de l’automate, a réussi à mettre l’automate à l’heure et à la date correcte.

3.4 Autres difficultés  
La plupart des problèmes que j’ai rencontrés, j’ai réussi à les résoudre grâce à l’aide de mes camarades de classe. Il était intéressant de partager nos points de vue sur les différentes situations. L’aide codesys m’a aussi été précieuse lors de nombreuses déclarations de variables. Je me suis aussi un peu aidé de l’intelligence artificielle quand c’était difficile de trouver mes réponses sur le site de codesys.

# Suivi

Pour le suivi, il nous a été demandé de remplir un journal de travail. J’ai rempli le journal à chaque fois que je travaillais sur le projet du module 2 ou lorsque je rédigeais ce rapport technique du module 1. Il permet assez facilement de connaître le nombre d’heures de travail fournies pour ce projet.

# Bilan

Au début, il m’était difficile d’imaginer réaliser un tel projet. Pourtant, ce projet m’a énormément plu, malgré quelques petites incompréhensions sur le sujet. J’ai su rebondir face aux difficultés et je trouve que je me suis bien débrouillé, surtout lorsque je repense à la situation où je ne savais presque rien sur les automatismes avant de commencer. Je suis vraiment satisfait du travail accompli au cours de ces nombreuses semaines.

Après réflexion, si je devais à nouveau réaliser ce même projet, je sais désormais comment je l’aborderais. Je l’élaborerais de manière plus sûre, plus spontanée, avec une logique différente de celle que j’ai utilisée pour le projet du module 2. J’aurais également dû prévoir davantage de temps pour la réalisation du projet, car le mode recette n’a pas été facile à concevoir, et encore moins à déboguer.

# Conclusion

J’ai trouvé le travail pour le module 1 et le module 2 très intéressant. Ces projets m’ont permis d’acquérir une grande quantité de connaissances, tant pour la rédaction de dossiers et de rapports que pour la programmation d’automates. Cela a été une expérience marquante, car cela m’a ouvert à un domaine totalement nouveau, surtout que je viens d’un univers quelque peu différent. J’ai pu comprendre plus en profondeur le fonctionnement d’une machine simple, et cette découverte m’a réellement passionné.

Ce travail a renforcé mon intérêt pour l’automatisation et la programmation industrielle. Non seulement j’ai appris à concevoir et à programmer un automate, mais j’ai aussi acquis des compétences en gestion de projet, en réflexion technique et en résolution de problèmes pratiques. Ce n’était pas facile, mais chaque étape m’a permis de repousser mes limites et de développer une vision plus complète de ce domaine. À travers ce projet, j’ai pris conscience de l’importance de la planification et de la rigueur dans le développement de systèmes automatisés, mais aussi de la satisfaction qu’on ressent lorsqu’on surmonte des défis complexes.

Je suis désormais plus confiant pour aborder des projets similaires à l’avenir, et j’ai hâte de continuer à évoluer dans cette direction.

# Source et annexes