REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE AMAR TELIDJI LAGHOUAT


Faculté de Technologie
Département- Électronique

# Mémoire de Master

**Domaine :** Sciences et Technologies
**Filière :** Automatique
**Option :** Automatique et informatique industrielle

EL-HOUITI Maria Kheira et CHIKHI Walid

**Thème**

# Automatisation d'un système de stockage/déstockage d'un magasin par PLC SIEMENS S7-1214

## Membres du jury :

Président : Pr Mohammed BELKHIRI
Examinateur : Dr Nabil ABOUCHABANA
Encadrant : Dr Aboubakeur HADJAISSA
Co-Encadrant : Dr Khaled AMEUR

2021/2022

# Remerciements

Louange à dieu le miséricordieux qui nous permis de bien accomplir ce modeste travail. Mes remerciements s'adressent tout particulièrement à notre encadreur Monsieur Hadj Aissa Aboubakar pour l'effort fourni, ses conseils scientifiques, sa patience et sa persévérance dans le suivi. Nos remerciements aussi à Monsieur Ameur Taki Eddin, qui nous a aidé énormément à faire notre travail. Aussi Mr Bouzidi Mohamed Redha pour ses conseils concernant la rédaction du mémoire.
Nous tenons a exprimé nos remerciements les plus sincère aux membres de jury Pr Mohammed Belkhiri et Dr Nabil Abouchabana et Dr Khaled AMEUR qui nous ont fait le très grand honneur de participer à l'amélioration de nos travaux.
Comme nous adressons également nous remerciements, à tous mes enseignants, qui nous ont donnés les bases de la science pendant les cinq ans de ma formation.
A toute personne qui a participé de près ou de loin pour l'accomplissement de ce modeste travail. Merci donc nos mères, nos pères, nos familles pour leurs présence, leurs confiance et leurs soutien.

# Abstract

The objective of this project is to design an automated system using a PLC, such system capable of controlling and supervising a warehouse, the studied system capable of storing and destocking pieces with considerable speed and has difficult places to reach. The programming of the specifications of this system was made in a PLC of the manufacturer SIEMENS (S7-1200) under the environment TIA Portal, also the different validation tests of this program are made in the same environment.

Thus, the development of a Human-Machine interface by the WinCC software that is integrated in the TIA Portal, in order to allow the operator to visualize and analyze the system study.

**Keywords** : TIA Poratl, PLC, S7-1200, WinCC.

## Résumé

L'objectif de ce projet est de concevoir un système automatisé à l'aide d'un API, tel système capable de contrôler et superviser un entrepôt, le système étudié apte de stocker et déstocker des pièces avec une rapidité considérable et a des endroits difficiles d'atteindre. La programmation de cahier des charge de ce système a été faite dans un API du constructeur SIEMENS (S7-1200) sous l'environnement TIA Portal, de même les différents testes de validation de ce programme sont faites dans le même environnement.

Ainsi, l'élaboration d'une interface Homme-Machine par le logiciel WinCC qui est intégré dans le TIA Portal, afin de permettre à l'opérateur de visualiser et analyser le système étudié.

**Mots clés** : TIA Poratl, API, S7-1200, WinCC.

**ملخص**

الهدف من مشروعنا هو تصميم نظام الي باستعمال API وهذا النظام هو عبارة عن نظام قادر على التحكم والاشراف على مراقبة مستودع النظام المدروس مؤهل ليقوم بتخزين تفريغ الحمولات بسرعة كبيرة مقارنة بأماكن يصعب الوصول اليها. برمجة احتياجات واختصاصات هذا النظام تم تجسيدها اعتمادا على API من الشركة المصنعة SIEMENS باستعمال البيئة TIA Portal مرورا بالاختبارات واعتماد صحة البرنامج.

بالإضافة الى اعداد واجهة HMI باستعمال البرنامج WinCC المدمج داخل TIA Portal لكي يستطيع المشغل من قراءة و تحليل النظام المدروس.

**الكلمات المفتاحية:** TIA Poratl, API, S7-1200, WinCC

# Table des matières

**Introduction générale** **1**
0.1 Problèmes et objectifs . . . 1
0.2 Organisation du manuscrit . . . 2

**1 Les systèmes automatisés** **3**
1.1 Introduction . . . 3
1.2 Initialisation aux APIs . . . 3
1.3 Principe et fonctionnement de l'API . . . 3
1.4 Architecture des APIs . . . 5
1.4.1 Module d'alimentation . . . 5
1.4.2 L'unité centrale de traitement (CPU) . . . 6
1.4.3 Interface entrées / sorties . . . 6
1.4.4 Console de programmation . . . 7
1.5 Types des APIs . . . 8
1.5.1 Type 1 : Le monobloc . . . 8
1.5.2 Type 2 : Le modulaire . . . 8
1.6 Les langages de programmation . . . 10
1.6.1 Langage de programmation LADDER . . . 10
1.6.2 Schéma fonctionnel (FBD) . . . 10
1.6.3 Langage ST (Texte Structuré) . . . 11
1.7 Qu'est-ce qu'une IHM . . . 11
1.7.1 Définition d'une IHM . . . 11
1.7.2 Utilisation d'une IHM . . . 12
1.8 Conclusion . . . 13

**2 Système de stockage / déstockage** **14**
2.1 Introduction . . . 14
2.2 Définition du Grafcet . . . 14
2.2.1 Mode de représentation . . . 14
2.2.2 Règles d'évolution d'un Grafcet . . . 15
2.3 Niveaux d'un Grafcet . . . 16
2.4 Système de stockage/déstockage . . . 17
2.4.1 Fonctionnement . . . 18
2.5 Schéma synoptique . . . 19

2.6 Capteurs et actionneurs . . . 19
2.6.1 Stockage . . . 19
2.6.2 Déstockage . . . 20
2.7 Grafcet du cahier de charges . . . 20
2.7.1 Grafcet niveau 1 . . . 20
2.7.2 Table d'activation et désactivation . . . 23
2.8 SIMATIC S7-1200 . . . 25
2.8.1 Critère de choix . . . 26
2.9 Conclusion . . . 26

**3 Simulation et validation** **27**
3.1 Introduction . . . 27
3.2 TIA Portal . . . 27
3.3 STEP 7 . . . 27
3.4 S7-PLCSIM . . . 28
3.5 WinCC . . . 28
3.6 Simulation du système de stockage . . . 28
3.6.1 Le convoyeur d'entrée . . . 28
3.6.2 Stockage du boîtier . . . 30
3.6.3 Remarques . . . 39
3.6.4 Déstockage du boîtier . . . 40
3.6.5 Remarque . . . 44
3.6.6 Convoyeur de sortie . . . 50
3.7 Création de dispositif IHM . . . 52
3.8 Les vues de supervision . . . 52
3.8.1 La vue du menu principal . . . 52
3.8.2 La vue du convoyeur d'entrée . . . 53
3.8.3 La vue du convoyeur de sortie . . . 53
3.8.4 La vue de stockage/déstockage . . . 53
3.8.5 La vue des emplacements . . . 54
3.8.6 La vue de la fourche . . . 54
3.9 Conclusion . . . 55

**Conclusion** **56**

# Table des figures


1.1 Automates SIEMENS. . . . . 4
1.2 Fonctionnement cyclique d'un API. . . . . 4
1.3 Architecture interne d'un API. . . . . 5
1.4 Architecture interne d'un API. . . . . 7
1.5 Console de programmation (supervision). . . . . 8
1.6 Un API de type monobloc. . . . . 9
1.7 Un API de type modulaire. . . . . 9
1.8 Un bloc fonctionnel. . . . . 11
1.9 Ecran IHM tactile. . . . . 12
1.10 L'utilisation d'une IHM dans un système SCADA. . . . . 12

2.1 Représentation d'une étape. . . . . 15
2.2 Représentation d'une action. . . . . 15
2.3 Représentation d'une transition. . . . . 16
2.4 Niveau 1 d'un Grafcet. . . . . 17
2.5 Niveau 2 d'un Grafcet. . . . . 17
2.6 Le schéma synoptique du système. . . . . 19
2.7 Le Grafcet du convoyeur d'entrée. . . . . 20
2.8 Le Grafcet du Stockage. . . . . 21
2.9 L'organigramme du bloc fonctionnel "Stockage". . . . . 22
2.10 La disposition des emplacements. . . . . 22
2.11 Le Grafcet du déstockage. . . . . 23
2.12 Le Grafcet du convoyeur de sortie. . . . . 23

3.1 Étape 1. . . . . 28
3.2 Etape 2. . . . . 29
3.3 Étape 3. . . . . 29
3.4 Étape 4. . . . . 30
3.5 Étape 5. . . . . 31
3.6 Étape 6. . . . . 32
3.7 Étape 7. . . . . 32
3.8 emplacement occupé. . . . . 33
3.9 Le choisit de l'emplacement 14. . . . . 33
3.10 DB3 (partie1). . . . . 34
3.11 DB3 (partie2). . . . . 35

3.12 movingX+ et movingZ+. . . . . 35
3.13 Le bloc de données DB17. . . . . 36
3.14 Le programme du bloc DB17. . . . . 36
3.15 Étape 8. . . . . 37
3.16 Étape 9. . . . . 38
3.17 Étape 10. . . . . 38
3.18 Étape 11. . . . . 39
3.19 movingX- et movingZ-. . . . . 40
3.20 Étape 30. . . . . 40
3.21 Le choisit de l'emplacement 16. . . . . 41
3.22 L'emplacement 16 est occupé. . . . . 41
3.23 DB8 (partie1). . . . . 42
3.24 DB8 (partie2). . . . . 42
3.25 movingX++ et movingZ++. . . . . 43
3.26 Étape 31. . . . . 44
3.27 Étape 32. . . . . 45
3.28 Étape 33. . . . . 46
3.29 Étape 34. . . . . 47
3.30 movingX– et movingZ–. . . . . 48
3.31 Étape 35. . . . . 48
3.32 Étape 36. . . . . 49
3.33 Étape 37. . . . . 49
3.34 Étape 38. . . . . 50
3.35 Étape 39. . . . . 51
3.36 Étape 40. . . . . 51
3.37 Liaison entre HMI et API. . . . . 52
3.38 La vue principale. . . . . 53
3.39 La vue du convoyeur d'entrée. . . . . 53
3.40 La vue du convoyeur de sortie. . . . . 54
3.41 La vue du système stockage. . . . . 54
3.42 La vue des emplacements. . . . . 55
3.43 La vue de la fourche. . . . . 55

# Liste des tableaux

1.1 Exemple des symboles les plus utilisés . . . . . 10

2.1 Capteurs et actionneurs du stockage . . . . . 19
2.2 Capteurs et actionneurs de déstockage . . . . . 20
2.3 Table d'activation/désactivation de stockage. . . . . 24
2.4 Tableau des actions associées (Stockage). . . . . 24
2.5 Table d'activation/désactivation de déstockage. . . . . 25
2.6 Tableau des actions associées (Déstockage). . . . . 25

# Liste des abréviations

**API** **A**utomate **P**rogrammable **I**ndustriel
**PLC** **P**rogrammable **L**ogic **C**ontroller
**IHM** **I**nterface **H**omme **M**achine
**SCADA** **S**upervisory **C**ontrol **A**nd **D**ata **A**cqisition
**IL** **I**nstruction **L**ist
**ST** **S**tructured **T**ext
**LD** **L**adder **D**iagram
**FBD** **F**unction **B**lock **D**iagram
**SFC** **S**equential **F**unction **C**hart
**GRAFCET** **GRA**phe **F**onctionnel de **C**ommande **E**tapes **T**ransitions
**CPU** **C**entral **P**rocessing **U**nit
**DC** **D**irect **C**urrent
**PID** **P**roportionnel **I**ntégral **D**érivé
**PC** **P**ersonal **C**omputer
**OPC** **O**pen **P**latform **C**ommunications
**ROM** **R**ead **O**nly **M**emory
**RAM** **R**andom **A**ccess **M**emory
**EEPROM** **E**lectrically **E**rasable **P**rogrammable **R**eads **O**nly **M**emory
**IEC** **I**nternational **E**lectrotechnical **C**ommission
**TIA Portal** **T**otally **I**ntegrated **A**utomation **P**ortal

# Introduction générale

Les systèmes automatisés font tellement partie de notre quotidien qu'on a pas toujours conscience de la technicité de ses dispositifs. Les usagers interagissent très naturellement avec des machines à laver, des portes de garage, des ascenseurs, des distributeurs de boissons ou bornes automatiques d'accueil.
Les lignes de production des usines modernes sont également équipées de systèmes automatisés capables d'accomplir de nombreuses taches, parfois très complexes. Les trains et les avions embarquent de nombreux systèmes automatisés qui facilitent leur pilotage. C'est également le cas de nos voitures actuelles dont le pilotage reste pour l'instant manuel, mais les constructeurs mettent actuellement au point des voitures autonomes à grands renforts du systèmes automatisés [8].

## 0.1 Problèmes et objectifs

On peut dire, d'une manière simplifiée, que l'automatisation industrielle est l'optimisation des processus industriels au moyen des systèmes automatisés, c'est-à-dire que l'utilisation de technologies dans processus spécifiques avec le but d'augmenter la productivité, l'autonomie, d'améliorer les conditions de travail et de simplifier certaines opérations pénibles.
Avec l'automatisation des processus de fabrication et la réduction de l'effort humain qui en résulte, on gagne plus de sécurité dans l'exécution des taches, car on peux éliminer ou simplifier les services considérés comme dangereux
[18].

Actuellement, lorsque nous entrons dans un espace de fabrication, ce que nous voyons se sont plusieurs machines produisant à un rythme constant. Nous avons également pu observer des personnes qui ne font qu'opérer, surveiller et entretenir ces équipements, grâce à l'utilisation de capteurs, d'actionneurs et de technologies telles que la robotique et les logiciels informatiques [5].

Les systèmes de l'automatisation remplissent différentes fonctions clés. Celles-ci sont, d'une part, le mesurage et le contrôle. Les autres tâches du système sont notamment la régulation et la communication. La salle de contrôle est une composante technique essentielle pour la commande des différents procédés. C'est ce qu'on appelle les interactions Homme-machines.
Les différents sous-domaines de la technique de l'automatisation incluent des composants techniques particuliers. Ainsi, le mesurage est effectué à l'aide de capteurs. Ces sondes déterminent les propriétés physiques et chimiques telles que l'humidité, la pression ou la chaleur. Le mesurage propose un vaste choix de capteurs, qui ont été spécialement conçus pour les systèmes automatisés.
La commande des installations en automatisation s'effectue au moyen d'un Automate Programmable Industriel (API) sur la base du numérique. Les ordres de contrôle parviennent au système par des actionneurs qui sont par exemple des moteurs, des vannes ou des aimants. Auparavant, les systèmes étaient composés d'automates à programmes câblés dont les modifications de programme nécessitaient une adaptation des câblages.

Pour l'API en revanche, seul le programme reste à modifier. Ce dispositif de commande a été développé en 1968 aux États-Unis et fut introduit en Europe en 1973 [2].
Dans ce contexte, l'objectif de ce projet est de faire une étude complète d'un cahier des charge industriel dont l'objectif d'automatiser et de superviser le fonctionnement d'un entrepôt, c'est à dire de stocker ou/et déstocker des pièces dans des endroits où l'être humain peut pas les atteindre. cette étude est passée par les étapes suivantes :

- Définition du cahier de charges.
- Définition des différents entrées/sorties du système étudié et leurs types.
- Modélisation du cahier de charges par l'outil Grafcet.
- Matérialisation du Grafcet obtenu .
- Choix de l'API.
- Programmation de l'API par l'un des cinq langages, dans notre cas : LADDER et ST .
- Test et validation de programme.
- Conception de système de supervision HMI.
- Élaboration d'une communication entre API et HMI, dans notre cas : PROFINET.
- Validation complète de fonctionnement de système étudié.

## 0.2 Organisation du manuscrit

Après la présentation d'introduction, le manuscrit comporte 3 chapitres et il est organisé comme suit :

Dans le premier chapitre, nous avons introduit les APIs et leur fonctionnement en citant les différents langages de programmation, de plus, nous avons également évoqué l'interface IHM.

Ensuite dans le deuxième chapitre, nous présentons le cahier de charge du système Stockage et Déstockage en utilisant l'outil de modélisation le "Grafcet".

Finalement dans le troisième chapitre, on va procéder à la programmation à base du logiciel TIA PORTAL STEP7/WinCC à l'aide du langage LADDER.

A la fin, on va conclure ce manuscrit par une conclusion générale et perspectives.

Chapitre 1

# Les systèmes automatisés

## 1.1 Introduction

Un automate programmable industriel (ou API) est un dispositif électronique programmable destiné à automatiser des processus sans l'intervention de l'homme. Il est désormais utilisé dans de nombreux secteurs d'activité, pour la commande des machines et des chaînes de production, la régulation de processus ou encore dans le secteur du bâtiment pour le contrôle de l'éclairage, du chauffage, de la sécurité... etc. Dans ce chapitre, nous allons décrire le principe de fonctionnement des automates programmables industriels, qui la clé importante de notre étude. Nous présentons les structures des APIs qu'ils composent de différents modules,les différents types existants et leurs principe de fonctionnement. A la fin de ce chapitre, nous allons abordé une aperçue sur le système de supervision qui est l'interfeçage entre la machine et l'être humain qui s'appeler HMI.

## 1.2 Initialisation aux APIs

Les APIs peuvent être considérés en terme simple comme des ordinateurs industriels dotés des interfaces d'entrées et de sorties et des fonctions spécialement conçues. Ils ont été initialement conçus pour remplacer les systèmes de relais électromécaniques afin d'offrir une solution plus simple pour modifier le fonctionnement d'un système de commande, plutôt que d'avoir à câbler une grande banque de relais, un chargement rapide à partir d'un PC ou d'un périphérique de programmation permet de modifier la logique de contrôle en quelques secondes.

Les APIs sont des constructions disponibles dans les langages de programmation pour permettre aux développeurs de créer plus facilement des fonctionnalités complexes(figure 1.1).

## 1.3 Principe et fonctionnement de l'API

L'automate programmable reçoit des données par ses entées, celles-ci sont ensuite traitées par un programme défini, le résultat obtenu étant délivré par ses sorties. Ce


FIGURE 1.1 – Automates SIEMENS.

cycle de traitement est toujours le même, quel que soit le programme, néanmoins le temps d'un cycle de l'API varie selon la taille du programme et la puissance de l'automate.

C'est l'unité centrale qui gère l'automate programmable : elle reçoit, mémorise et traite les données entrantes et détermine l'état des données sortantes en fonction du programme établi(figure 1.2).

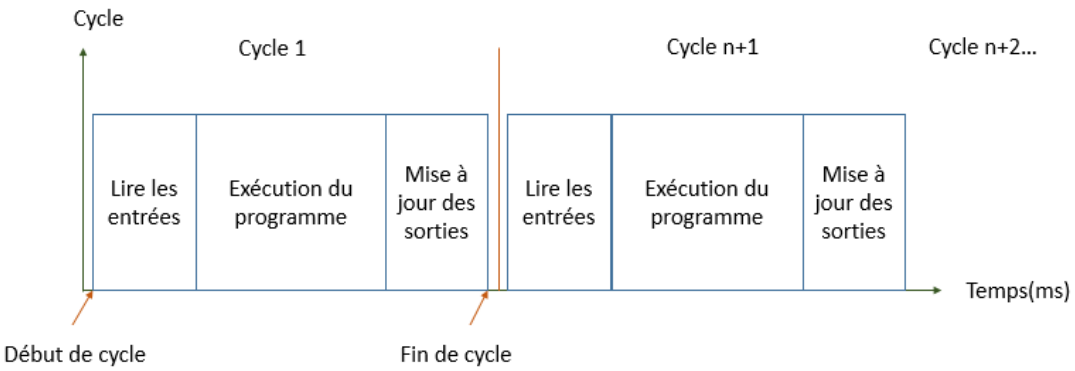



FIGURE 1.2 – Fonctionnement cyclique d'un API.

## 1.4 Architecture des APIs

Les APIs comportent cinq principaux modules arrangés l'un à côté de l'autre(figure 1.3), tels que [6] :

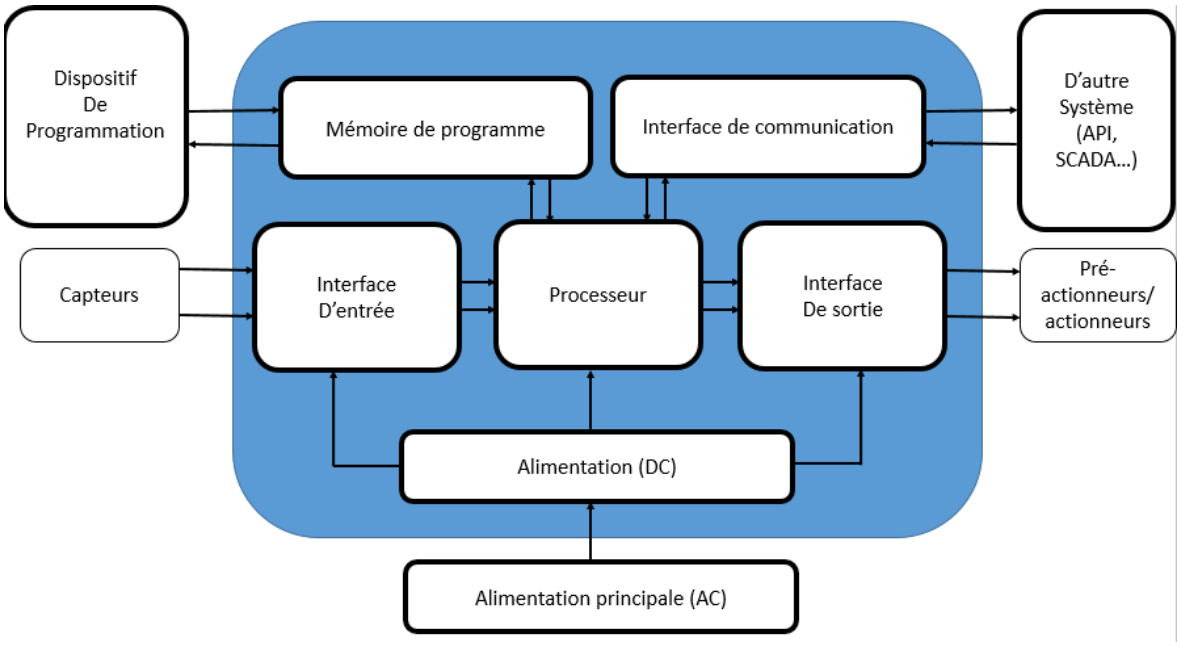



FIGURE 1.3 – Architecture interne d'un API.

— Une alimentation DC.
— Une unité centrale (CPU) à base de microprocesseur doté d'une carte mémoire.
— Interfaces d'entrées.
— Interface de sorties.
— Des interfaces de communication.

### 1.4.1 Module d'alimentation

Ce module est utilisé pour fournir la puissance requise à l'ensemble du système API. Il convertit la tension d'entrées alternative (220V) du secteur en une tension continue (24V) nécessaire au processeur et aux circuits des modules d'interface d'E/S. Les API peuvent être divisés en deux types principaux en fonction de leur alimentation : ceux avec une alimentation intégrée et ceux qui nécessitent une alimentation externe.

#### API sans alimentation intégrée

Ce type des API doivent être connectés à une source d'alimentation externe telle que des batteries.

#### API avec alimentation intégrée

Ce type a une tension intégrée DC 24V ou DC 48V, il ne sera donc jamais besoin d'une batterie ou d'un transformateur séparé.

### 1.4.2 L'unité centrale de traitement (CPU)

Le module CPU est le cerveau de l'API contenant le microprocesseur. Cette unité interprète les signaux d'entrées et exécute les actions de commande en fonction du programme enregistré dans sa mémoire, communiquant les décisions sous forme des signaux d'actions aux sorties. Ce module contient un interface de programmation afin de communiquer avec le console de programmation suivant un protocole bien déterminé (TCP/IP , MPI-bus ... etc).
La mémoire ROM est la mémoire ou est stocké le langage de programmation. Elle est en général figée, c'est-à-dire en lecture seulement, elle comprend un système d'exploitation, des pilotes et des programmes d'application. La mémoire RAM est la mémoire accessible en lecture-écriture pendant le fonctionnement. Elle est utilisée pour stocker des programmes et des données, généralement c'est une mémoire ROM effaçable électriquement (EEPROM) d'une capacité varie du 4 KO jusqu'au 50 KO.

### 1.4.3 Interface entrées / sorties

Les dispositifs d'entrée peuvent être des boutons poussoirs de démarrage et d'arrêt, des interrupteurs... etc., et les dispositifs de sortie peuvent être des vannes, des relais... etc.
Le module d'E/S permet d'interfacer les dispositifs d'entrée et de sortie avec le microprocesseur en recevant des informations de périphériques externes (capteurs) et de les communiqués aux périphériques internes (pré-actionneurs et actionneurs).Il y a deux types d'E/S, type Tout ou Rien (DI/DO) et analogique (AI/AO).
Plus de ces modules, on trouve des modules spéciaux d'E/S (carte PID, carte de comptage rapide...), ce type des cartes dotés des microprocesseurs afin de simplifier les taches et soulager le module CPU(figure 1.4).

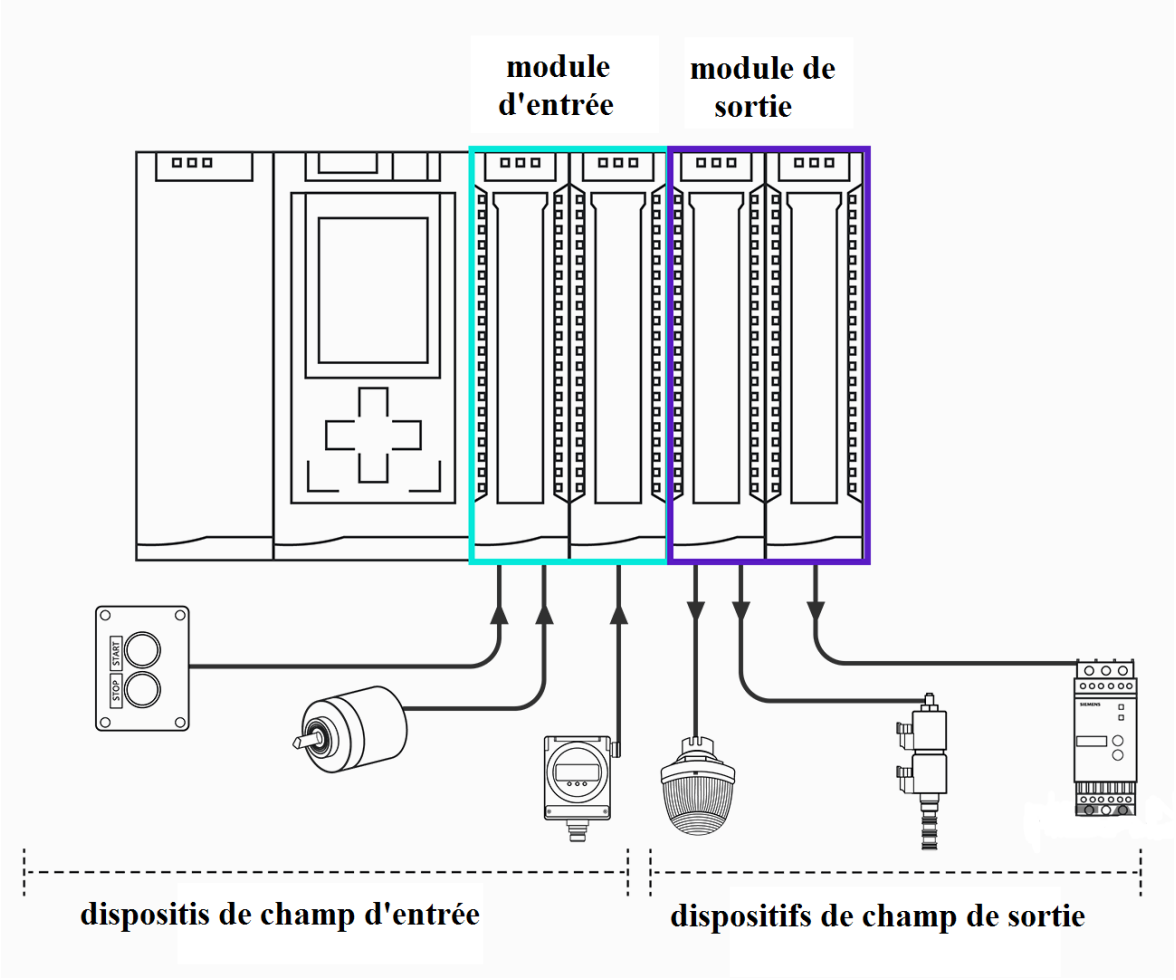



FIGURE 1.4 – Architecture interne d'un API.

### 1.4.4 Console de programmation

Ce module est utilisé pour recevoir et transmettre des données sur des réseaux de communication depuis ou vers d'autres systèmes distants tels que API, SCADA, HMI, Serveur OPC...etc.
Il permet la vérification du périphériques, l'acquisition de données, la synchronisation entre les systèmes et la gestion de la connexion.

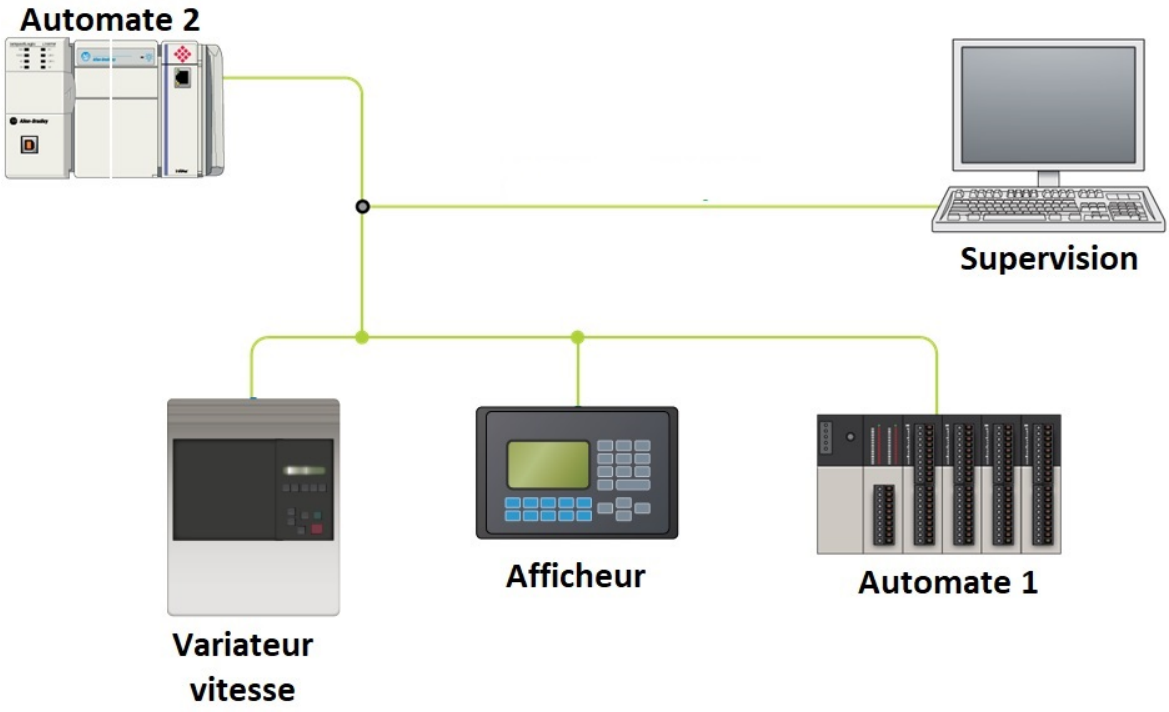



FIGURE 1.5 – Console de programmation (supervision).

## 1.5 Types des APIs

Quand il s'agit les types des APIs, on distingue deux types selon le nombre d'E/S [15] :

### 1.5.1 Type 1 : Le monobloc

On peut dire aussi c'est l'API fixe, petit et déterminé par le fabriquant car les sections d'entrées et de sorties sont intégrées dans le microprocesseur lui-même. Cela signifie que le nombre d'entrées et de sorties ne peut pas être augmenté dans ce type d'API(figure 1.6). Généralement, il est destiné aux applications simples, il se caractérise par un coût faible et moins complexe.

### 1.5.2 Type 2 : Le modulaire

Un API modulaire est un type qui permet de multiples extensions du système API grâce à l'utilisation de modules, d'où le terme « modulaire ». Ces modules donnent à l'automate programmable des fonctionnalités supplémentaires comme le nombre accru d'interface E/S. Le module d'alimentation, de communication, d'entrées et de sorties sont tous séparés du microprocesseur, donc ils doivent être connectés manuellement afin de créer le système de contrôle. La propriété d'extension le rend important en termes de nombre d'E/S plus cher et complexe(figure 1.7).


FIGURE 1.6 – Un API de type monobloc.


FIGURE 1.7 – Un API de type modulaire.

## 1.6 Les langages de programmation

Selon la norme IEC 1131-3, il existe deux classifications principales des langages de programmation d'API, sont ensuite divisés en de nombreux types sous-classes [11] :

— Langages textuels :

  — IL : Liste d'instructions.

  — ST : Texte structuré.

— Langages graphiques :

  — LD : Schéma à contacts (Ladder).

  — FBD : Schéma fonctionnel.

  — SFC : Sequential Function Chart.

Bien que tous ces langages de programmation API soient définis afin de structurer l'organisation interne des programmes et des blocs fonctionnels. Généralement, les langages graphiques sont préférés aux langages textuels.

### 1.6.1 Langage de programmation LADDER

Le schéma à contacts (ou langage LADDER) est la forme la plus simple et populaire pour programmer les API. C'est un langage proche des schémas électriques en représentant des fonctions logiques (AND, OR...).
Le langage LADDER est un langage visuel qui fournit des confirmations d'état pour la plupart des instructions, ce qui permet de connaître facilement le parcours du programme et de comprendre la logique [9].

| | |
|---|---|
| -\| \|- | variable d'entrée ou contact à fermeture |
| -\|/\|- | variable d'entrée complimentée ou contact à ouverture |
| -( )- | variable de sortie |
| -(S)- | sortie mise à un mémorisée (S=set) |
| -(R)- | sortie mise à zéro mémorisée (R=reset) |
| \| | connexions verticales |

TABLE 1.1 – Exemple des symboles les plus utilisés

### 1.6.2 Schéma fonctionnel (FBD)

C'est une méthode simple et graphique pour programmer plusieurs fonctions entre les variables d'entrées et de sorties sous forme d'un réseau de blocs élémentaires.
L'avantage d'utiliser le FBD est le pouvoir d'ajouter n'importe quel nombre d'E/S en connectant la sortie d'un bloc de fonction à l'entrée d'un autre, par lequel la construction d'un schéma fonctionnel [10](figure 1.8).

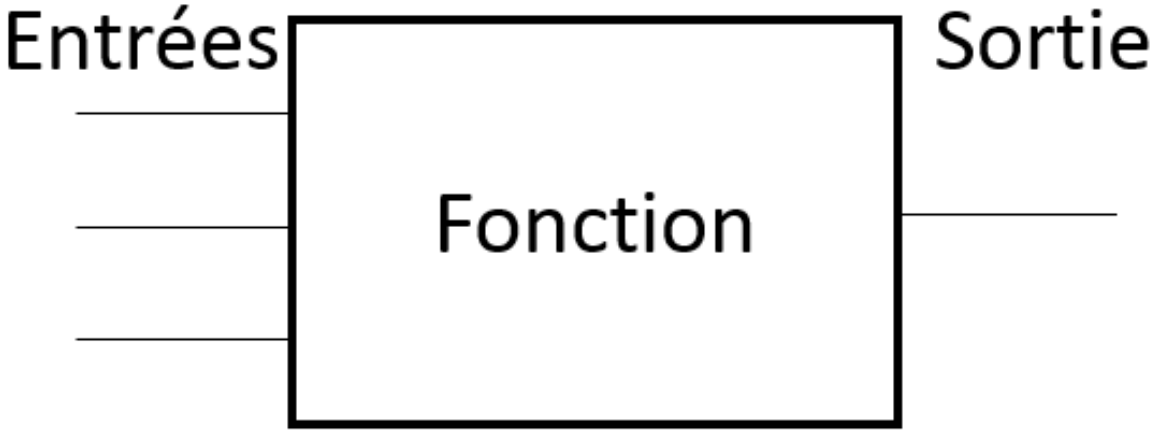



FIGURE 1.8 – Un bloc fonctionnel.

### 1.6.3 Langage ST (Texte Structuré)

C'est un langage de programmation qui ressemble beaucoup au langage C ou assembleur. Ceci offre une transition simple en API pour ce qui a un fond dans un langage de programmation traditionnel.
Tout simplement,l'utilisateur entre des instructions codées qui s'exécutent d'une façon séquentielle, en évaluant des fonctions spécifiques, vérifiant les booléens d'un API [10].

## 1.7 Qu'est-ce qu'une IHM

Aujourd'hui, les systèmes de contrôle industriels poursuit de se transformer progressivement et de développer dans le monde, les taches que les opérateurs doivent accomplir peuvent changer fréquemment.
Afin de gérer cette complexité, les contrôles doivent être flexibles en visant l'amélioration et l'accroissement de la productivité. C'est l'avantage de l'IHM.[16]

### 1.7.1 Définition d'une IHM

IHM signifie Interface Homme-Machine, c'est une interface qui permet à un utilisateur de communiquer avec une machine, un système ou à un appareil. Le terme IHM peut être appliqué à n'importe quel écran utilisé pour interagir avec un appareil, mais généralement il est plus couramment utilisé dans le contexte d'un processus industriel. Les IHM affichent des données en temps réel et permettent à l'utilisateur de contrôler les machines grâce à une interface utilisateur graphique (figure 1.9).[7]

L'interfeçage fait appel aux trois principales fonctions d'interactions humaines :

— Le toucher (commande par boutons,écrans tactiles,claviers,pavés numériques).

— La vision (Surveillance et contrôle sur écran).

— L'écoute (Alarmes sonores, Bips).


FIGURE 1.9 – Ecran IHM tactile.

### 1.7.2 Utilisation d'une IHM

Quand un système de contrôle et d'acquisition de données (SCADA) communique avec les automates programmables, et les capteurs d'entrée et de sortie pour obtenir des informations sur le fonctionnement des équipements, ces informations sont affichées sur une IHM.
Une IHM peut afficher ces informations sous forme de graphique, de tableau ou d'autre représentation visuelle qui facilite la lecture et la compréhension. Les opérateurs peuvent également voir et gérer les alarmes à l'aide d'une IHM, cela leur permet d'être sûrs de pouvoir réagir rapidement(figure 1.10).[16]

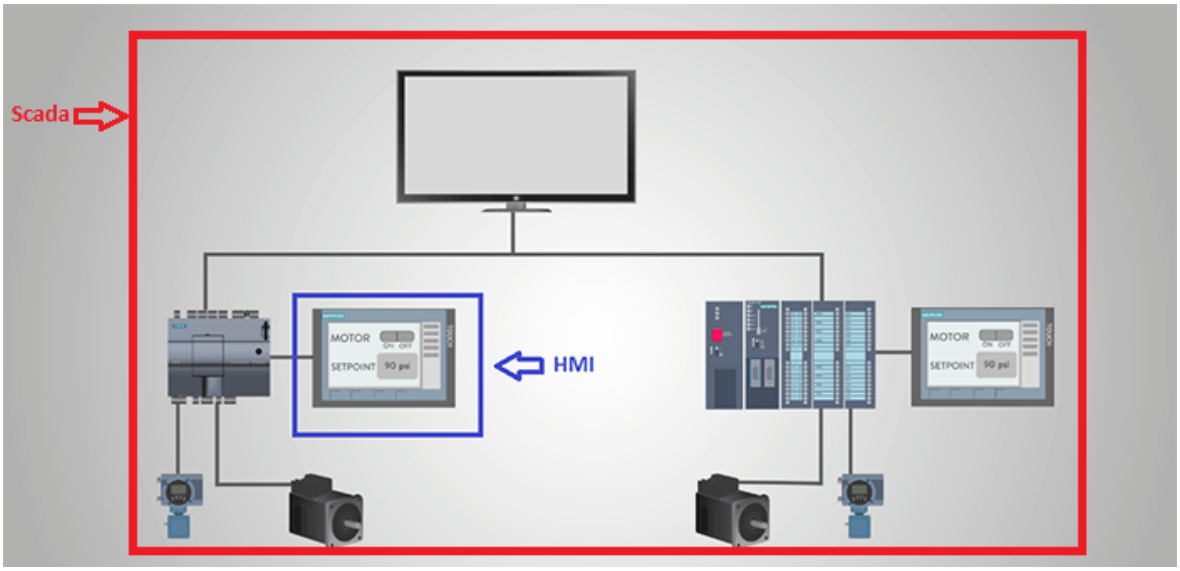



FIGURE 1.10 – L'utilisation d'une IHM dans un système SCADA.

## 1.8 Conclusion

L'objectif de ce chapitre était de parler sur les APIs et de donner une idée en ce qui concerne son domaine d'utilisation et son principe de fonctionnement,à la fin de ce chapitre on a essayer de donner une bref aperçue sur le système de supervision (IHM).

Chapitre 2

# Système de stockage / déstockage

## 2.1 Introduction

Pour que le fonctionnement d'un système automatisé soit analysé et bien compris, il doit être modélisé Il est donc essentiel que les méthodes et les outils utilisés permettent de véhiculer les informations élaborées au long du déroulement du processus. Pour cela, on va décrire le fonctionnement de notre système automatisé.
Dans ce chapitre, nous présenterons le cahier de charge de stockage et dé-stockage d'un entrepôt, en détaillant ses différentes étapes par le diagramme fonctionnel (GRAFCET).

## 2.2 Définition du Grafcet

Le Grafcet est un mode de représentation et d'analyse d'un automatisme, particulièrement bien adapté aux systèmes à évolution séquentielle, c'est-à-dire décomposable en étapes.C'est un outil graphique et puissant, directement exploitable, car c'est aussi un langage pour la plupart des API existants sur le marché.[14]
Le Grafcet est donc un langage graphique représentant le fonctionnement d'un automatisme par un ensemble :

— d'étapes auxquelles sont associées des actions.
— de transitions entre étapes auxquelles sont associées des conditions de transition.
— des liaisons orientées entre les étapes et les transitions.

### 2.2.1 Mode de représentation

Nous décrivons ici la représentation selon la norme EN 60848 :

#### Étape

Une étape symbolise un état du système automatisé. L'étape possède deux états possibles : active représentée par un jeton dans l'étape ou inactive. L'étape i, représentée

par un carré repéré numériquement, possède ainsi une variable d'état, appelée variable d'étape Xi. Cette variable est une variable booléenne valant 1 si l'étape est active, sinon 0.
La situation initiale d'un système automatisé est indiquée par une étape dite étape initiale et représentée par un carré double[4].(figure 2.1)

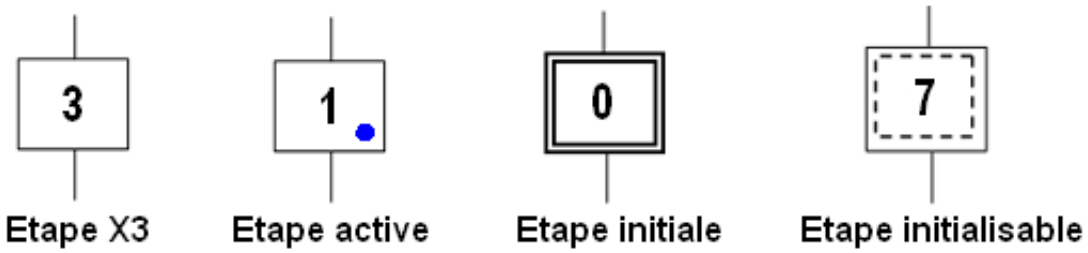



FIGURE 2.1 – Représentation d'une étape.

**Actions associées aux étapes**

A chaque étape est associée une action ou plusieurs, c'est à dire un ordre vers la partie opérative ou vers d'autres Grafcets. Mais on peut rencontrer aussi une même action associée à plusieurs étapes ou une étape vide (sans action)[4].(figure 2.2)

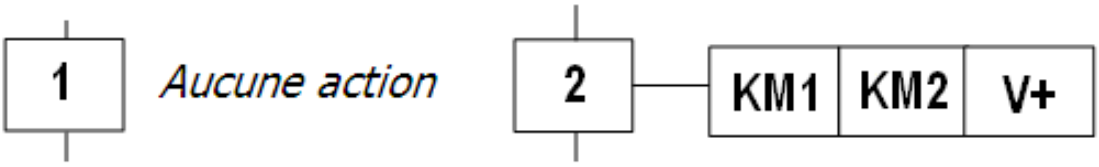



FIGURE 2.2 – Représentation d'une action.

**Transition**

Une transition est une condition de passage d'une étape à une autre. Elle n'est que logique (dans son sens Vrai ou Faux) et sans notion de durée. A chaque transition est associée une réceptivité.(figure 2.3)
La réceptivité est une condition logique qui doit être remplis pour le franchissement de la transition[4].

### 2.2.2 Règles d'évolution d'un Grafcet

Selon la norme EN 61131-3 ou selon la norme EN 60848, Le Grafcet possède un comportement dynamique dirigé par cinq règles [4] :

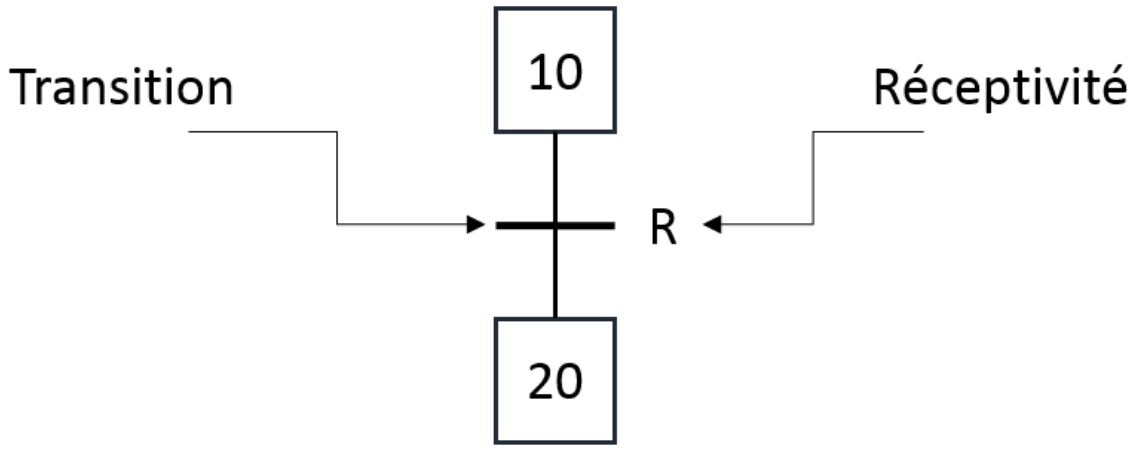



FIGURE 2.3 – Représentation d'une transition.

**Règle N°1 : Condition initiale**

A l'instant initial, seules les étapes initiales sont actives.

**Règle N°2 : Franchissement d'une transition**

Pour qu'une transition soit validée, il faut que toutes ses étapes amont (immédiatement précédentes reliées à cette transition) soient actives.
Le franchissement d'une transition se produit lorsque la transition est validée, ET seulement si la réceptivité associée est vraie.

**Règle N°3 : Évolution des étapes actives**

Le franchissement d'une transition entraîne obligatoirement l'activation de toutes les étapes immédiatement suivantes et la dés-activation de toutes les étapes immédiatement précédentes.

**Règle N°4 : Franchissement simultané**

Toutes les transitions simultanément franchissables à un instant donné sont simultanément franchies.

**Règle N°5 : Conflit d'activation**

Si une étape doit être simultanément désactivée par le franchissement d'une transition aval, et activée par le franchissement d'une transition amont, alors elle reste active.

## 2.3 Niveaux d'un Grafcet

— **Gracet niveau 1 :**
Dans ce Grafcet le système sera décrit sous forme littérale, sans tenir compte de

la technologie utilisée. Il est souvent utilisé pour vendre ou décrire un système. Il est l'outil idéal pour expliquer un système à des non professionnels ou établir un cahier des charges(figure 2.4).[1]

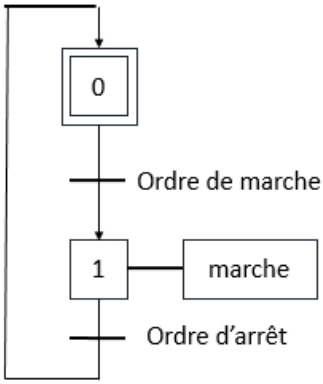



FIGURE 2.4 – Niveau 1 d'un Grafcet.

— **Grafcet niveau 2 :**
Ce niveau fournit une description technologique et opérationnelle du procédé et s'adresse à des spécialistes(figure 2.5).

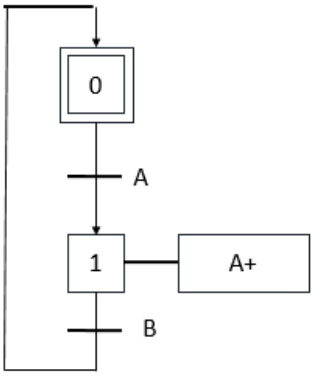



FIGURE 2.5 – Niveau 2 d'un Grafcet.

## 2.4 Système de stockage/déstockage

Notre projet est constitué d'un entrepôt, où il sera le processus de stockage et de déstockage des boîtiers. L'entrepôt s'agit d'une armoire commune où en mettant les pièces ou bien les boites stockées et aussi les faire évacuer. Le stockage se fait par une fourche de transport qui sort vers gauche et droite grâce à un moteur à deux sens de rotation.

Tout d'abord, le boîtier doit être situé sur le convoyeur d'entrée, et qu'il sera détecté par un capteur d'entrée et cela provoque la marche du convoyeur vers le capteur du chargement. L'arrivée du boîtier ici est le début de stockage dont la fourche le retire et attend l'ordre de position de destination où le boîtier sera stocké.
C'est le même cas pour le déstockage, cependant, au cours de ce processus il fait la vérification de l'emplacement désiré s'il est occupé il fait le déstockage, puis la fourche met la boite sur le convoyeur de sortie qui se termine par son évacuation. Tandis que, dans le stockage si l'emplacement est vide, il stocke le boîtier.
Le stockage et le déstockage sont deux processus indépendants qui utilisent une armoire commune.

### 2.4.1 Fonctionnement

Le fonctionnement de système en question est décrite en deux sous-systèmes :

**Stockage**

- Le démarrage du procédé est fait par le bouton "start".
- Quand l'action "Emitter" se réalise, le boîtier sera sur le convoyeur qui va être détecté par le capteur "At entry" sans oublier que le convoyeur ne contient aucune boite.
- Le convoyeur commence à marcher jusqu'à l'arrivée du boîtier au capteur "At load".
- La fourche sort vers gauche "Forks left" et se détecte par "At left" qui va porter la boite "Lift" et revient au milieu "At middle".
- La fourche sera orienté vers la place de stockage par deux moteurs, le 1er passe horizontalement sur l'axe X vers gauche ou droite et le 2ème passe verticalement vers haut ou bas.
- La fourche sort encore une fois vers droite "Forks right" et se détecte par "At right" pour stocker le boîtier et retourne au point de départ.

**Déstockage**

- Le démarrage se fait par le même bouton "start".
- La fourche attend l'ordre d'orientation vers l'emplacement désiré.
- On choisit l'emplacement qu'on va vider et avec la même façon, la fourche se déplace au niveau de l'axe X et Z par les actionneurs "Moving X" et "Moving Z".
- La fourche sort vers gauche pour récupérer la boite "Lift" et retourne au "At middle" après à sa position initiale afin de la décharger à travers le convoyeur.
- Le capteur "At unload" détecte la présence du boîtier et provoque la marche du convoyeur vers le capteur "At exit" et de faire l'évacuation par "Remover".

## 2.5 Schéma synoptique

langage Le schéma synoptique du système est présenté dans la figure(figure 2.6).

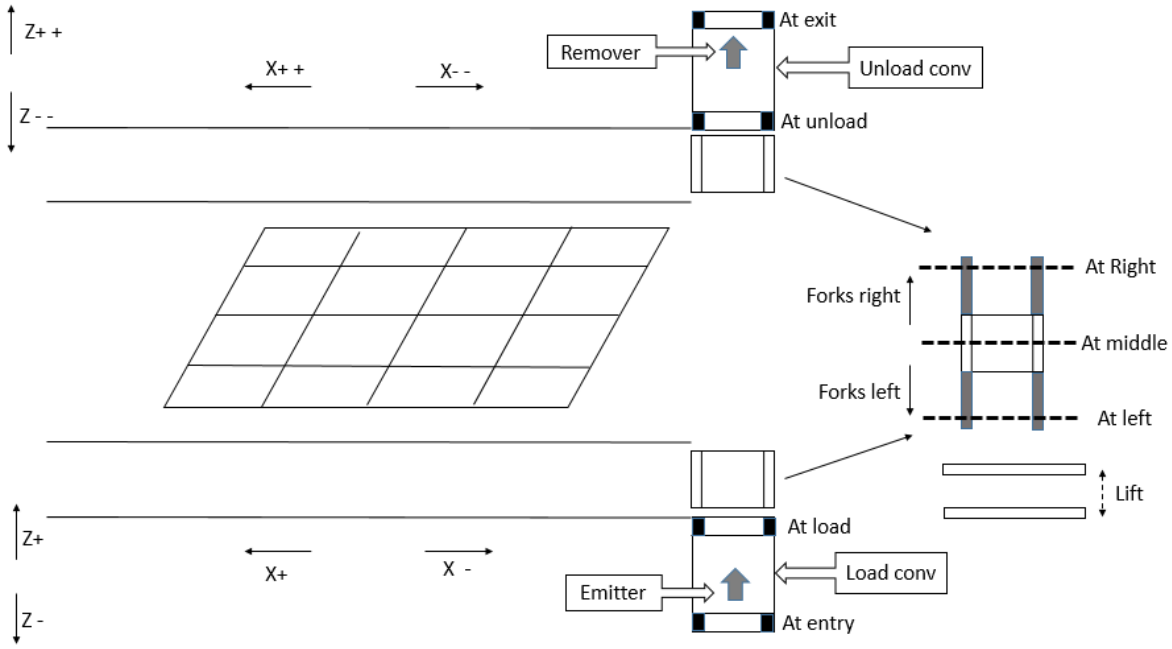



FIGURE 2.6 – Le schéma synoptique du système.

## 2.6 Capteurs et actionneurs

Les capteurs qu'on a utilisé se sont de type de capteur de position, et la actionneurs sont constitués par des moteurs à deux sens de rotation.

### 2.6.1 Stockage

Le tableau suivant2.1 représente les actionneurs et les capteurs utilisés dans le stockage.

| Capteurs | Actionneurs |
|---|---|
| At entry | Forks left s |
| At left s | Forks right s |
| At load | Lift s |
| At middle s | Emitter |
| At right s | Load conv |
| | moving X+ |
| | moving X- |
| | moving Z+ |
| | moving Z- |

TABLE 2.1 – Capteurs et actionneurs du stockage

### 2.6.2 Déstockage

Le tableau suivant 2.2 représente les actionneurs et les capteurs utilisés dans le déstockage.

| Capteurs | Actionneurs |
|---|---|
| At unload | Forks left u |
| At exit | Forks right u |
| At right u | Lift u |
| At middle u | Remover |
| At left u | Unload conv |
| | moving X++ |
| | moving X– |
| | moving Z++ |
| | moving Z– |

TABLE 2.2 – Capteurs et actionneurs de déstockage

## 2.7 Grafcet du cahier de charges

### 2.7.1 Grafcet niveau 1

#### Stockage

Le chargement du boîtier se fait suivant la Grafcet 2.7) :

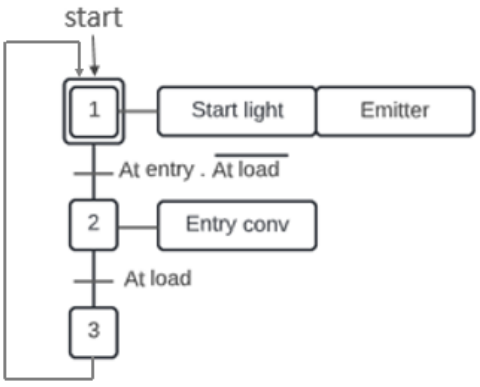



FIGURE 2.7 – Le Grafcet du convoyeur d'entrée.

Le stockage du boîtier se fait suivant le Grafcet 2.8) :

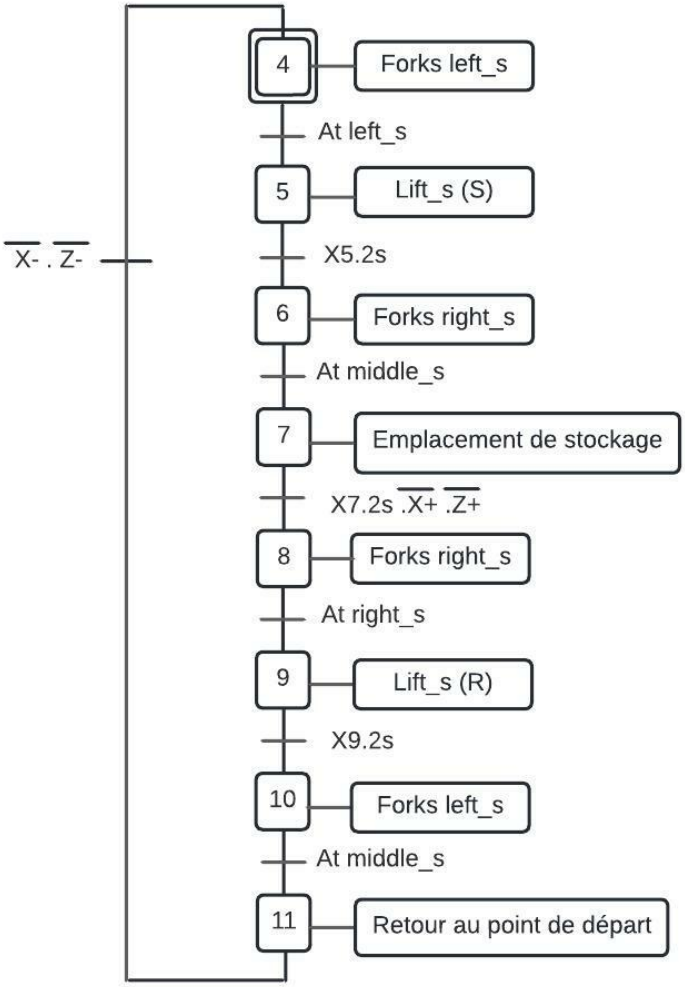



FIGURE 2.8 – Le Grafcet du Stockage.

L'action associée en étape 7 "Emplacement de stockage" est programmée dans un bloc fonctionnel en utilisant le langage ST. L'organigramme du programme sera représenté dans la figure 2.9).

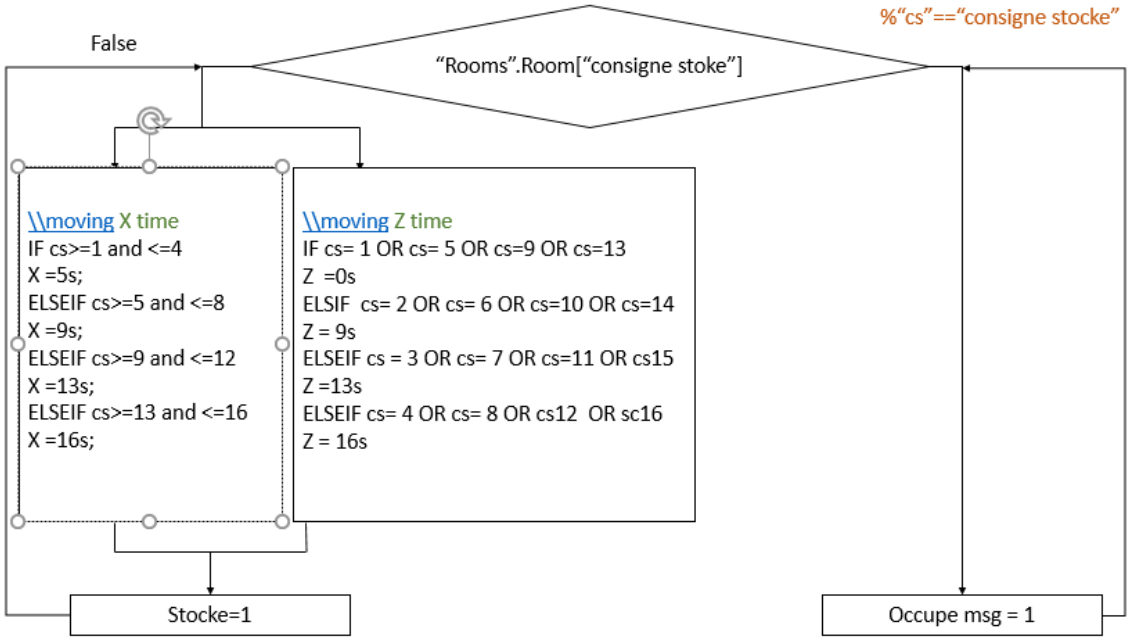



FIGURE 2.9 – L'organigramme du bloc fonctionnel "Stockage".

Les conditions du temporisation sont définies pour qu'elles s'adaptent avec La disposition des emplacements 2.10).

| | | | |
|---|---|---|---|
| Emplacement 4 "Rooms".Room[4] | Emplacement 8 "Rooms".Room[8] | Emplacement 12 "Rooms".Room[12] | Emplacement 16 "Rooms".Room[16] |
| Emplacement 3 "Rooms".Room[3] | Emplacement 7 "Rooms".Room[7] | Emplacement 11 "Rooms".Room[11] | Emplacement 15 "Rooms".Room[15] |
| Emplacement 2 "Rooms".Room[2] | Emplacement 6 "Rooms".Room[6] | Emplacement 10 "Rooms".Room[10] | Emplacement 14 "Rooms".Room[14] |
| Emplacement 1 "Rooms".Room[1] | Emplacement 5 "Rooms".Room[5] | Emplacement 9 "Rooms".Room[9] | Emplacement 13 "Rooms".Room[13] |

FIGURE 2.10 – La disposition des emplacements.

### Déstockage

Le déstockage du boîtier se fait suivant le Grafcet 2.11) :

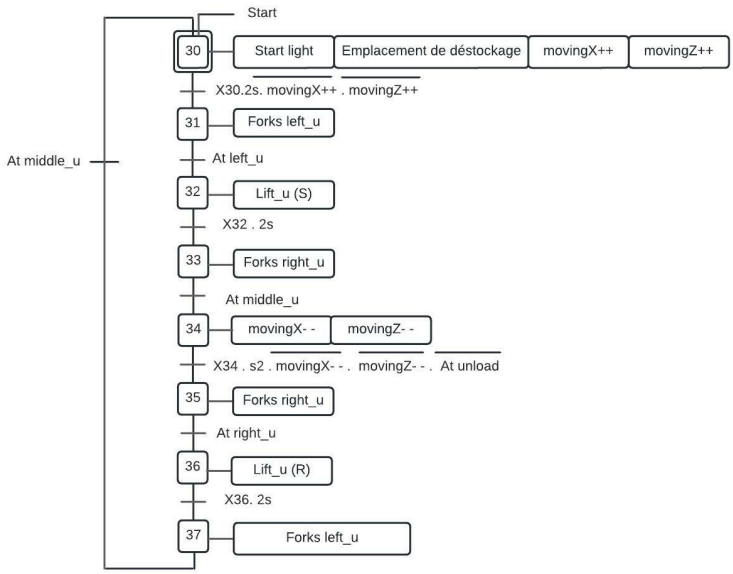



FIGURE 2.11 – Le Grafcet du déstockage.

Le déchargement du boîtier se fait suivant la Grafcet 2.12) :

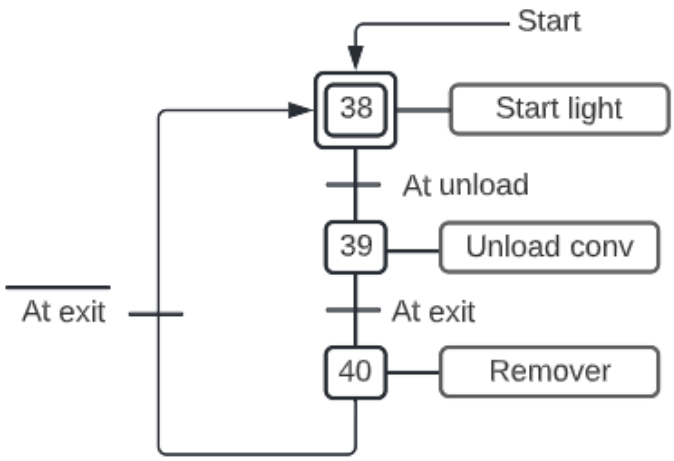



FIGURE 2.12 – Le Grafcet du convoyeur de sortie.

### 2.7.2 Table d'activation et désactivation

#### Stockage

Le tableau suivant 2.3 représente les conditions d'activation et de désactivation de chaque étape du Grafcet du système stockage :

| Xi | CAXi | CDXi |
|---|---|---|
| X1 | Start | X2+ESD |
| X2 | At entry . $\overline{Atload}.X1$ | X3+ESD |
| X3 | At load . X2 | X4+ESD |
| X4 | At load. $\overline{X5}.\overline{X6}.\overline{X7}.\overline{X8}.\overline{X9}.\overline{X10}.\overline{X11} + X11.\overline{movingX-}.\overline{movingZ-}$ | X5+ESD |
| X5 | X4 . At left s | X6+ESD |
| X6 | X5 . 4s | X7+ESD |
| X7 | X6 . At middle s + empty msg . X7 | X8+ESD |
| X8 | stocke . 2s . $\overline{movingX+}.\overline{movingZ+}$ | X9+ESD |
| X9 | X8 . At right s | X10+ESD |
| X10 | X9 . 2s | X11+ESD |
| X11 | X10 . At middle s | X4+ESD |

TABLE 2.3 – Table d'activation/désactivation de stockage.

Le tableau suivant 2.4 représente les actions associées aux étapes :

| Xi | Actions |
|---|---|
| X1 | Emitter + Start light |
| X2 | Entry conv |
| X4 | Forks left s |
| X5 | Lift (S) |
| X6 | Forks right s |
| X7 | Stockage + movingX+ + movingZ+ |
| X8 | Forks right s + occupe msg (R) |
| X9 | Lift (R) |
| X10 | Forks left s |
| X11 | movingX- + movingZ- |

TABLE 2.4 – Tableau des actions associées (Stockage).

**Déstockage**

Le tableau suivant 2.5 représente les conditions d'activation et de désactivation de chaque étape du Grafcet du système déstockage :

| Xi | CAXi | CDXi |
|---|---|---|
| X30 | emptymsg.X30+X37.At middleu+Start.$\overline{X31}.\overline{X32}.\overline{X33}.\overline{X34}.\overline{X35}.\overline{X36}.\overline{X37}$ | X31+ESD |
| X31 | X30 . 2s .$\overline{movingX++}.\overline{movingZ++}$ | X32+ESD |
| X32 | X31 . At left u | X33+ESD |
| X33 | X32 . 2s | X34+ESD |
| X34 | X33 . At middle u | X35+ESD |
| X35 | X34 . 2s . $\overline{movingX--}.\overline{movingZ--}.\overline{Atunload}$ | X36+ESD |
| X36 | X35 . At right u | X37+ESD |
| X37 | X36 . 2s | X30+ESD |
| X38 | Start+ X40 . $\overline{Atexit}$ | X39+ESD |
| X39 | X38 . At unload | X40+ESD |
| X40 | X39 . At exit | X38+ESD |

TABLE 2.5 – Table d'activation/désactivation de déstockage.

Le tableau suivant 2.6 représente les actions associées aux étapes :

| Xi | Actions |
|---|---|
| X30 | Start light + Déstockage + movingX++ + movingZ++ |
| X31 | Forks left u |
| X32 | Lift u (S) |
| X33 | Forks right u |
| X34 | movingX- - + movingZ- - |
| X35 | Forks right u |
| X36 | Lift u (R) |
| X37 | Forks left u |
| X38 | start light |
| X39 | Unload conv |
| X40 | Remover |

TABLE 2.6 – Tableau des actions associées (Déstockage).

## 2.8 SIMATIC S7-1200

L'automate programmable S7-1200 offre la flexibilité et la capacité de contrôler une grande variété d'appareils pour diverses tâches d'automatisation. Le CPU intègre un microprocesseur, et une alimentation intégrée, ainsi que d'autres éléments dans le boîtier compact, tels que des circuits d'entrée et de sortie. Il permet l'utilisation de différentes logiques de programmation telles que les instructions de comptage, la synchronisation, les opérations mathématiques complexes ou la logique booléenne, Ceci permet à l'automate de lire les entrées et de pouvoir les modifier sur la sortie.
Dans le modèle S7-1200, il existe différentes CPU, dans notre cas c'est : le 1214C DC/DC/DC de référence 6ES7214-1AG40-0XB0 avec la version de firmware V4.2, Notez que tous les capteurs connectés à ce CPU doivent être connectés au 24V DC.

### 2.8.1 Critère de choix

Plusieurs critères existent pour le choix d'un API, en citant :

— Nombre d'entrées / sorties : Possibilités d'extension.

— Type de processeur : la taille mémoire, la vitesse de traitement et les fonctions spéciales offertes par le processeur permettront le choix dans la gamme souvent très étendue.

— Fonctions ou modules spéciaux : certaines cartes (commande d'axe, pesage ...) permettront de "soulager" le processeur et devront offrir les caractéristiques souhaitées (résolution, ...).

— Simplicité de programmation qui offre un langage destiné à l'automaticien suivant la norme IEC 61131.

## 2.9 Conclusion

Nous avons modélisé le procédé de commande à l'aide du Grafcet. Nous avons élaboré en premier lieu un Grafcet de niveau 1 pour expliquer le système, puis le Grafcet de niveau 2 qui met en oeuvre et décrit les états des entrées et des sorties.
Au terme de ce chapitre, nous concluons que le Grafcet est un outil de modélisation qui permet facilement le passage d'un cahier de charges fonctionnel à un langage d'implémentation optionnel. Et cela nous permettra d'aborder la programmation qui pilotera le procédé au chapitre suivant à l'aide du STEP7.

Chapitre 3

# Simulation et validation

## 3.1 Introduction

Après l'élaboration du programme de commande de notre système à automatiser, nous arrivons à l'étape décisive du travail effectué. Cette étape est la validation du programme par simulation et vérification de son bon fonctionnement, pour cela nous avons utilisé le logiciel S7 PLCSIM de l'environnement TIA Portal . L'application de simulation de modules S7-PLCSIM nous permet d'exécuter et de tester notre programme dans un automate programmable virtuel [1].

Et comme deuxième point à élaboré, c'est la conception du système de supervision du système étudier à l'aide du logiciel WINCC de l'environnement TIA Portal.

## 3.2 TIA Portal

La plateforme TIA Portal ou bien Totally Integrated Automation Portal, est un environnement de développement tout en un permettant de programmer non seulement des automates mais aussi des afficheurs industriels (IHM). Il permet de mettre en œuvre des solutions d'automatisation avec un système d'ingénierie intégré comprenant les logiciels SIMATIC STEP 7 et SIMATIC WinCC [17].

## 3.3 STEP 7

STEP 7 est un logiciel d'ingénierie de Siemens qui permet de programmer des automates de la gamme Siemens. La nouvelle version de STEP 7 est fournit dans le logiciel de Siemens TIA Portal.
Le STEP 7 offre plusieurs fonctions pour l'automatisation d'une installation, en citant :

- Configuration et paramétrage du matériel.
- Programmation.
- Paramétrage de la communication.
- Test, mise en service et maintenance.
- Fonctions de diagnostic et d'exploitation.

Le logiciel STEP 7 de base pour les étudiants comporte un logiciel optionnel S7-PLCSIM.[13]

## 3.4 S7-PLCSIM

S7-PLCSIM offre une interface simple au programme utilisateur STEP 7 servant à visualiser et à modifier différents objets tels que les variables d'entrée et de sortie. Il permet le test dynamique des programmes de toute configuration automate SIMATIC S7 sans disposer du matériel cible [3].

## 3.5 WinCC

WinCC est un logiciel de Siemens permettant de créer des interfaces homme-machine sur pupitre tactile (IHM) ou sur écran. La dernière version de WinCC est intégrée à TIA Portal, permettant de familiariser avec le logiciel WinCC [12].
Les pupitres opérateur SIMATIC HMI sont de plus petit micro panel jusqu'au multi panel ainsi que d'un logiciel "runtime" [19].

## 3.6 Simulation du système de stockage

### 3.6.1 Le convoyeur d'entrée

La Grafcet du convoyeur d'entrée s'agit 3 étapes, dans la 1re étape on fait le démarrage du procédé en mettant la boite sur le convoyeur "Emitter" , la figure 3.1 représente l'étape 1 en langage LADDER.

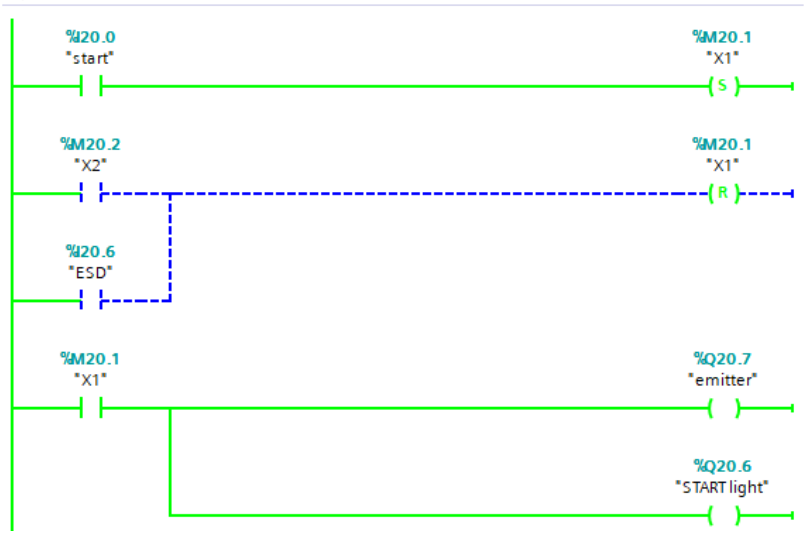



FIGURE 3.1 – Étape 1.

Dans la 2ème étape, le capteur "At entry" détecte la présence du boîtier ainsi que le convoyeur ne doit pas contenir aucune boite. Ce qui active l'étape 2, l'action associée consiste à marcher le convoyeur "Load conv" jusqu'au le capteur "At load". La figure 3.2 représente l'étape 2 en langage LADDER.

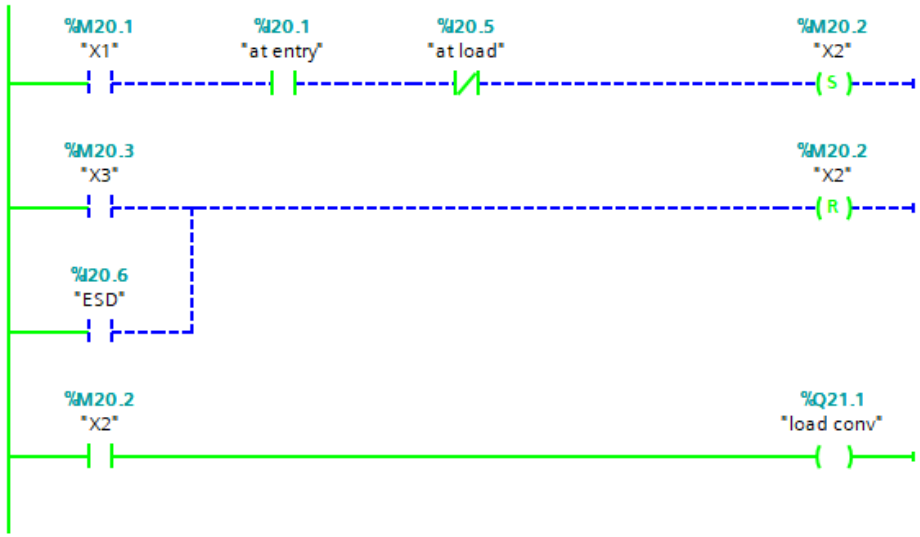



FIGURE 3.2 – Etape 2.

Dans la 3ème étape, Le boîtier a été chargé et prêt pour le stocker. La figure 3.3 représente l'étape 3 en langage LADDER.

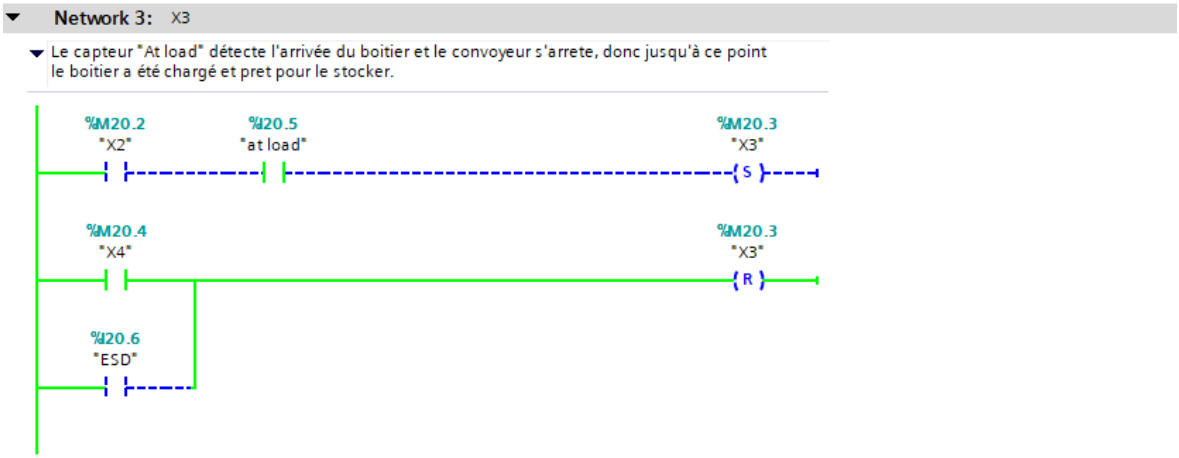



FIGURE 3.3 – Étape 3.

### 3.6.2 Stockage du boîtier

Le Grafcet du stockage s'agit 8 étapes ( de 4 jusqu'à 11), donc la 1re étape dans le bloc fonctionnel "stockage" est X4. Quand le capteur "At load" détecte la boite, l'étape 4 s'active, son action associée est la sortie de la fourche vers gauche en direction de la boite. La figure 3.4 représente l'étape 4 en langage LADDER.

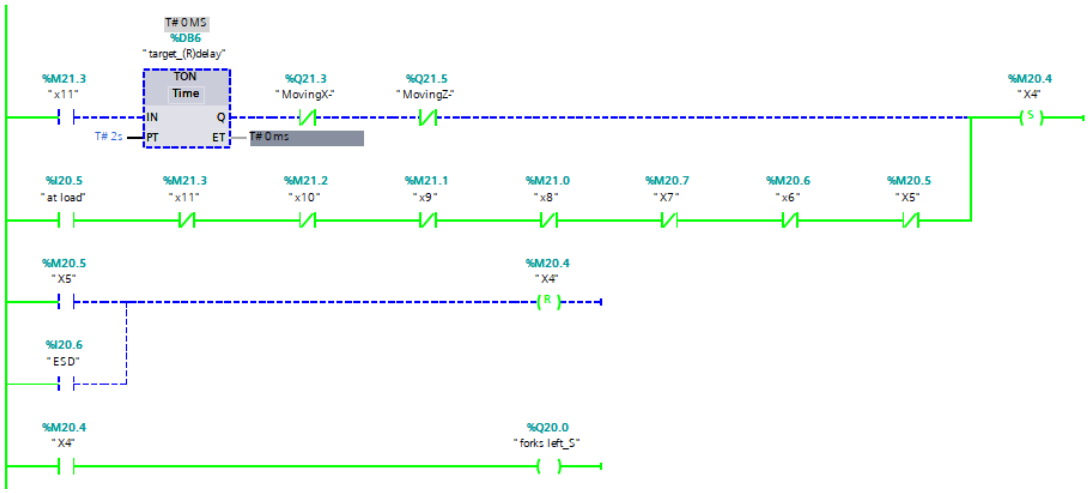



FIGURE 3.4 – Étape 4.

Dans l'étape 5, quand la fourche arrive à gauche, il y aura une détection au niveau du capteur "At left-s". Ce qui active l'étape actuelle et faire lever le boîtier par la fourche "Lift-s(SET)". La figure 3.5 représente l'étape 5 en langage LADDER.

Après 4s de l'étape précédente, l'étape 6 s'active et la fourche retourne au milieu "Forks right-s" , en attendant l'ordre de l'emplacement du stockage. La figure 3.6 représente l'étape 6 en langage LADDER.

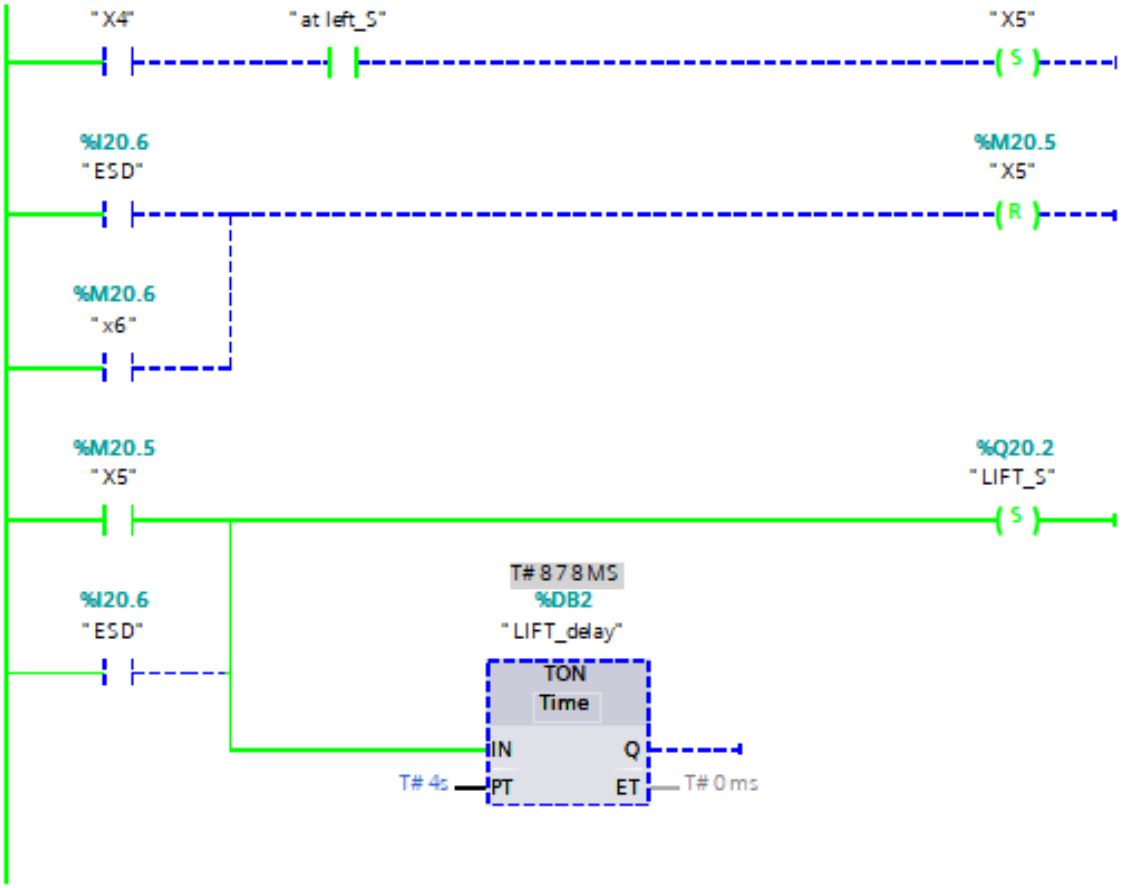



FIGURE 3.5 – Étape 5.

L'étape 7 représente le bloc d'emplacement du stockage, dont la fourche est au milieu "At middle-s". L'activation de cette étape accède au bloc des données (DB3) qui montre les conditions de marche de la fourche au niveau de l'axe X (horizontalement) et Z (verticalement) et les instructions à suivre. La figure 3.7 représente l'étape 7 en langage LADDER.

L'étape 7 peut être activée par une 2ème branche qui est : "**occupe msg . X7**". Cette branche a un rôle important en cas de l'emplacement est occupé. La figure 3.8 montre ça.

Dans ce cas, on a choisit l'emplacement 14 pour stocker la boite dedans. L'emplacement désiré s'écrit dans le champ de la variable "consigne stock", cela est représenté dans la figure 3.9.

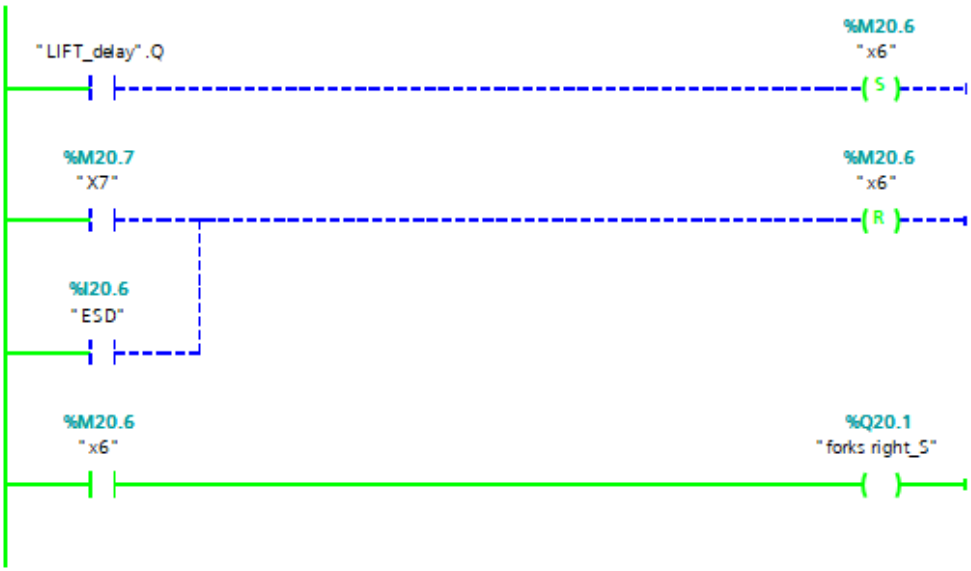



FIGURE 3.6 – Étape 6.

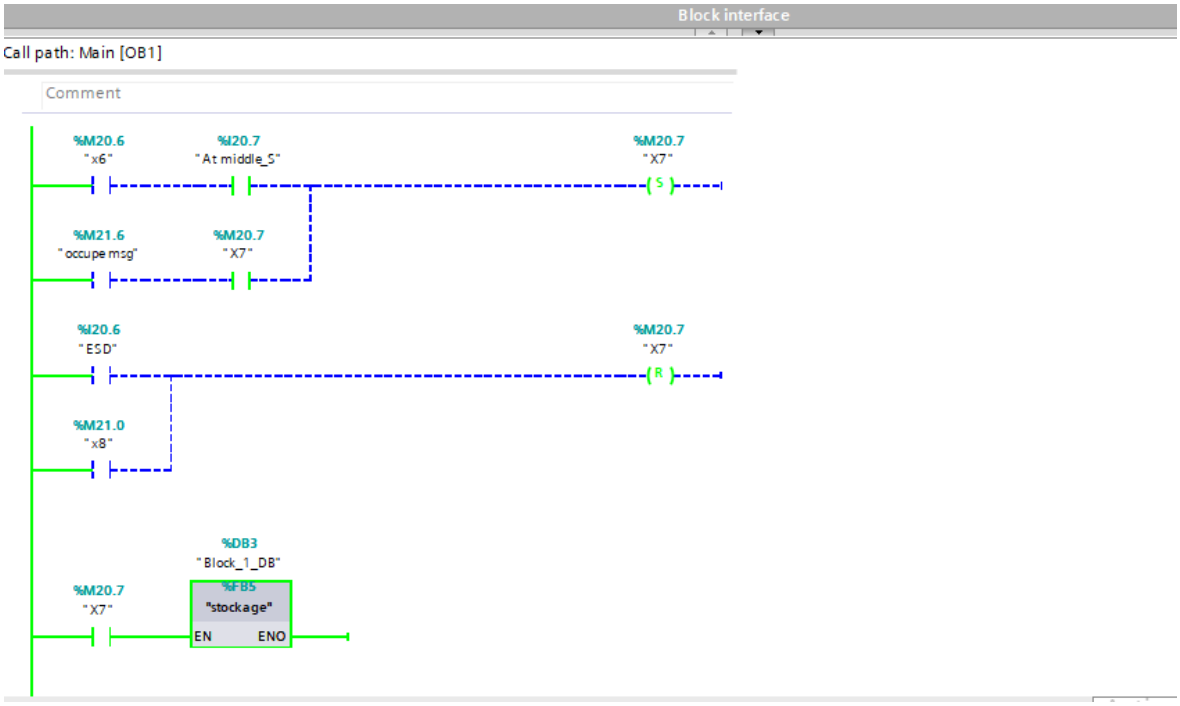



FIGURE 3.7 – Étape 7.

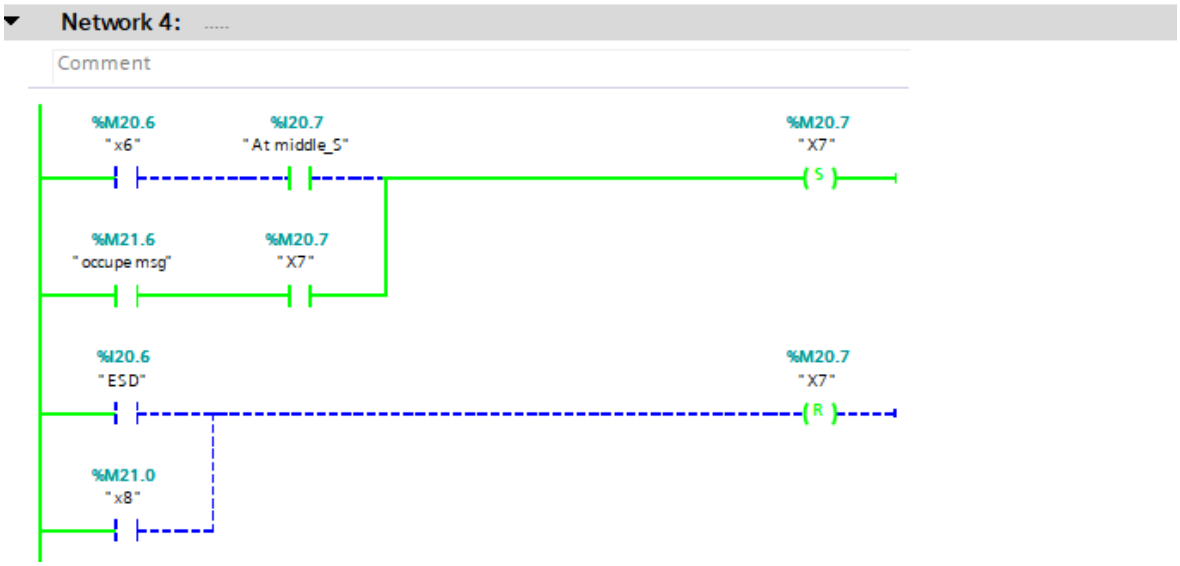
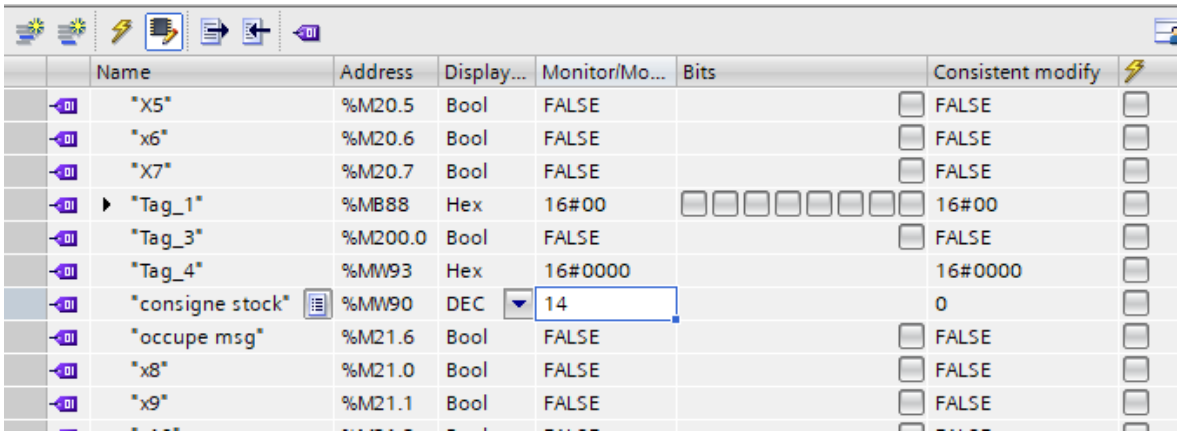



FIGURE 3.8 – emplacement occupé.



| | Name | Address | Display... | Monitor/Mo... | Bits | Consistent modify |
|---|---|---|---|---|---|---|
| | "X5" | %M20.5 | Bool | FALSE | | FALSE |
| | "x6" | %M20.6 | Bool | FALSE | | FALSE |
| | "X7" | %M20.7 | Bool | FALSE | | FALSE |
| | "Tag_1" | %MB88 | Hex | 16#00 | | 16#00 |
| | "Tag_3" | %M200.0 | Bool | FALSE | | FALSE |
| | "Tag_4" | %MW93 | Hex | 16#0000 | | 16#0000 |
| | "consigne stock" | %MW90 | DEC | 14 | | 0 |
| | "occupe msg" | %M21.6 | Bool | FALSE | | FALSE |
| | "x8" | %M21.0 | Bool | FALSE | | FALSE |
| | "x9" | %M21.1 | Bool | FALSE | | FALSE |

FIGURE 3.9 – Le choisit de l'emplacement 14.

Les figures 3.10 et 3.11 représente le bloc des données "stockage" écrit en langage ST (Texte structuré).

```
0001   IF "Rooms".Room["consigne stock"] = FALSE //l'emplacement est vide
0002   THEN
0003     // X moving time
0004     IF "consigne stock">=1 AND "consigne stock"<=4
0005   THEN
0006     //"Tag_4" := "consigne stock";
0007     "x" := T#5s;
0008   ELSIF "consigne stock">=5 AND "consigne stock"<=8
0009   THEN
0010     "x" := T#9s;
0011   ELSIF "consigne stock">=9 AND "consigne stock"<=12
0012   THEN
0013     "x" := T#13s;
0014   ELSIF "consigne stock">=13 AND "consigne stock"<=16
0015   THEN
0016     "x" := T#16s;
0017   END_IF;
0018
```

FIGURE 3.10 – DB3 (partie1).

Donc, d'après le DB3 la fourche se déplace sur l'axe X durant 16s et avec une durée de 9s sur l'axe Z. Cela est représenté dans la figure 3.12.

```
0019   // Z moving time
0020   IF "consigne stock" = 1 OR "consigne stock" = 5 OR "consigne
     stock"=9 OR "consigne stock"=13
0021   THEN
0022     //"Tag_4" := "consigne stock";
0023     "z" := T#0s;
0024   ELSIF  "consigne stock" = 2 OR "consigne stock" = 6 OR "consigne
     stock"=10 OR "consigne stock"=14
0025   THEN
0026     "z" := T#9s;
0027   ELSIF  "consigne stock" = 3 OR "consigne stock" = 7 OR "consigne
     stock"=11 OR"consigne stock"=15
0028   THEN
0029     "z" := T#13s;
0030   ELSIF  "consigne stock" = 4 OR "consigne stock" = 8 OR "consigne
     stock"=12  OR "consigne stock"=16
0031   THEN
0032     "z" := T#16s;
0033   END_IF;
0034   "stocke" := TRUE; //l'ordre de stockage
0035   ELSE
0036       "occupe msg" := TRUE; //l'emplacement est occupé
```

FIGURE 3.11 – DB3 (partie2).

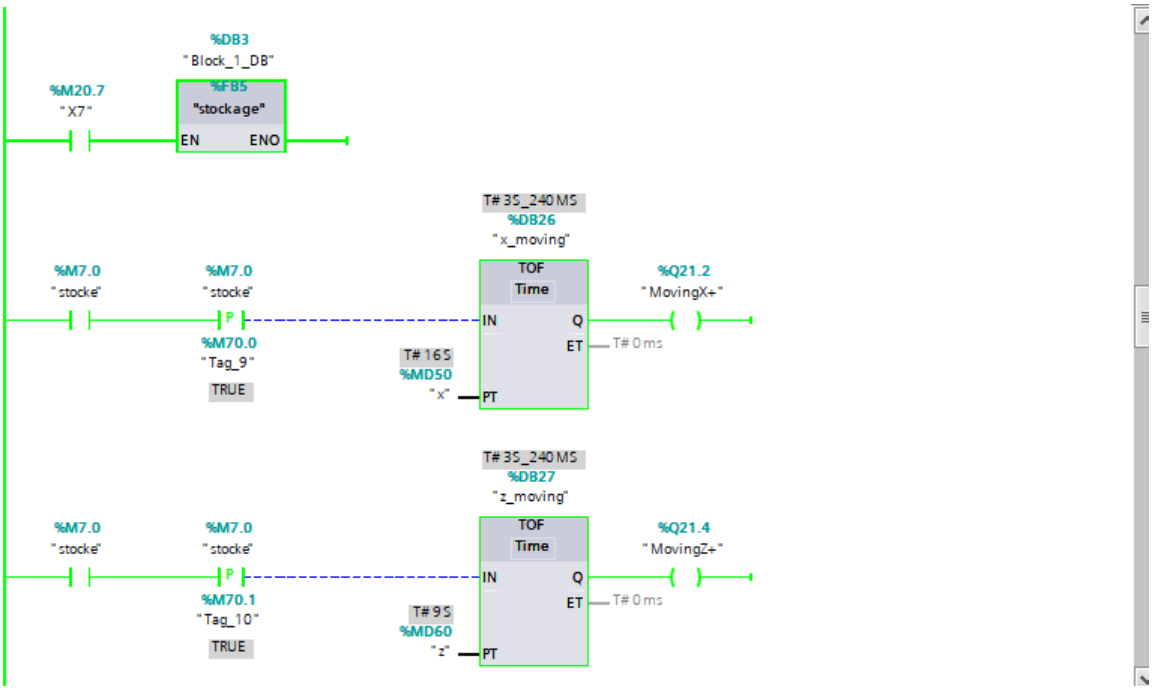



FIGURE 3.12 – movingX+ et movingZ+.

Dans l'étape 7 et avant de passer à l'étape 8, on passe par un bloc de données très essentiel, c'est le bloc DB17 (set/reset) 3.13qui fait la modification de l'état des emplacements après chacun des deux processus. Après le stockage l'emplacement reçoit un 1 et un 0 après le déstockage, alors l'état plein est représenté avec un 1 et l'état vide est représenté par un 0. La figure 3.14 représente le code du programme du bloc.

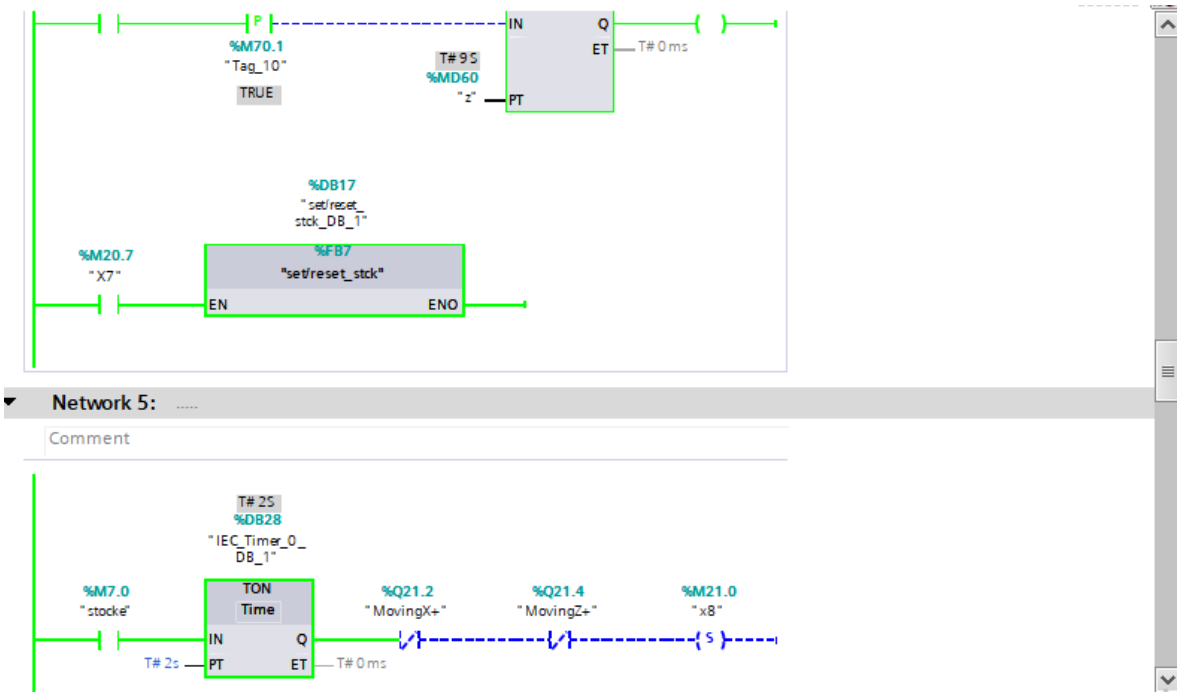



FIGURE 3.13 – Le bloc de données DB17.

```
1
2    IF "X7" = TRUE THEN
3        // Statement section IF
4        "Rooms".Room["consigne stock"] :=TRUE;
5    ELSIF "x34"=TRUE
6    THEN
7        "Rooms".Room["consigne unstock"] := FALSE;
8    END_IF;
9
```

FIGURE 3.14 – Le programme du bloc DB17.

Quand la fourche arrive à l'emplacement désiré, et les actionneurs "movingX+" et "movingZ+" s'arrêtent, l'étape 8 s'active et la fourche sort à droite pour entrer le boîtier. La figure 3.15 représente l'étape 8 en langage LADDER.

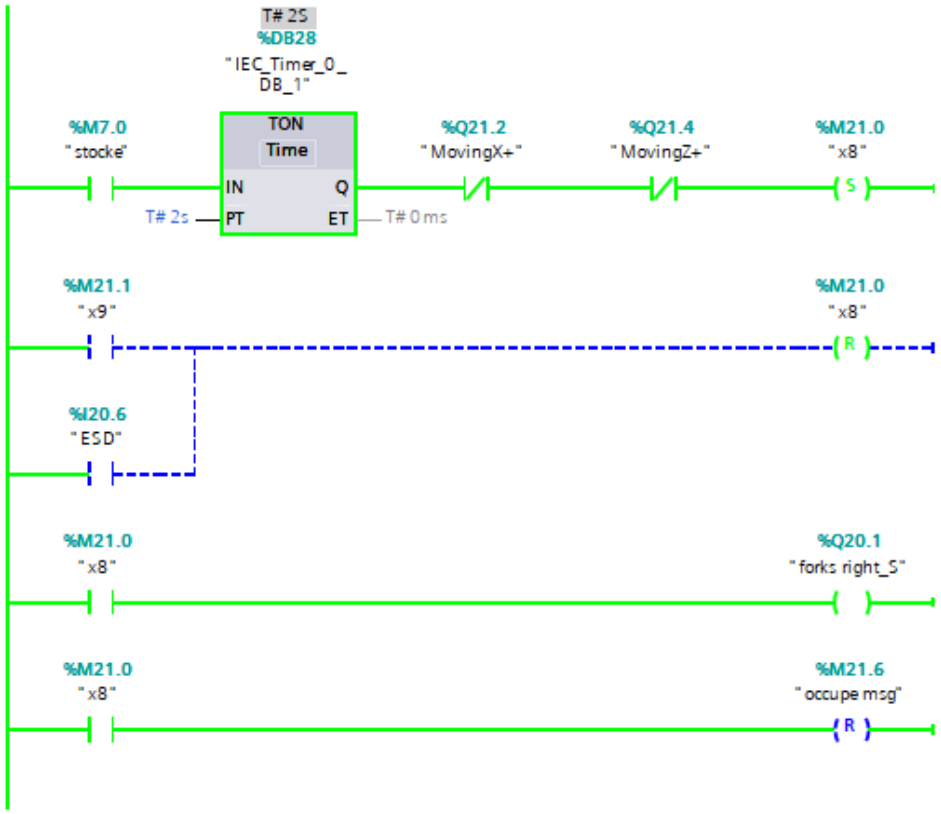



FIGURE 3.15 – Étape 8.

Dans l'étape 9, le capteur "At right-s" détecte la fourche et s'abaisse et mets la boite. La figure 3.16 représente l'étape 9 en langage LADDER.

Dans l'étape 10, la fourche retourne au milieu "Forks left-s". La figure 3.17 représente l'étape 10 en langage LADDER.

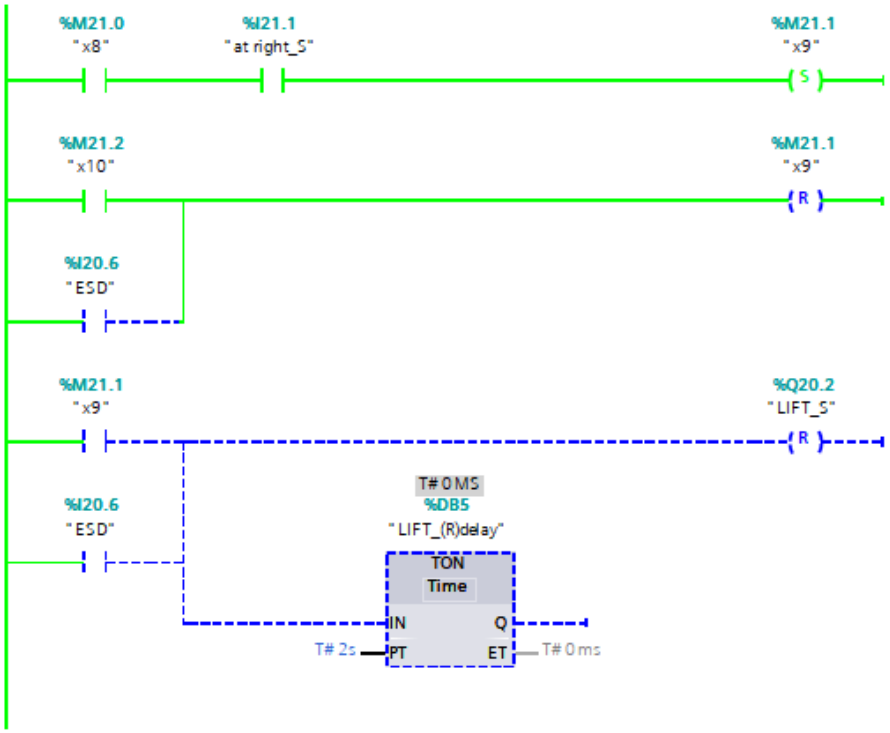



FIGURE 3.16 – Étape 9.

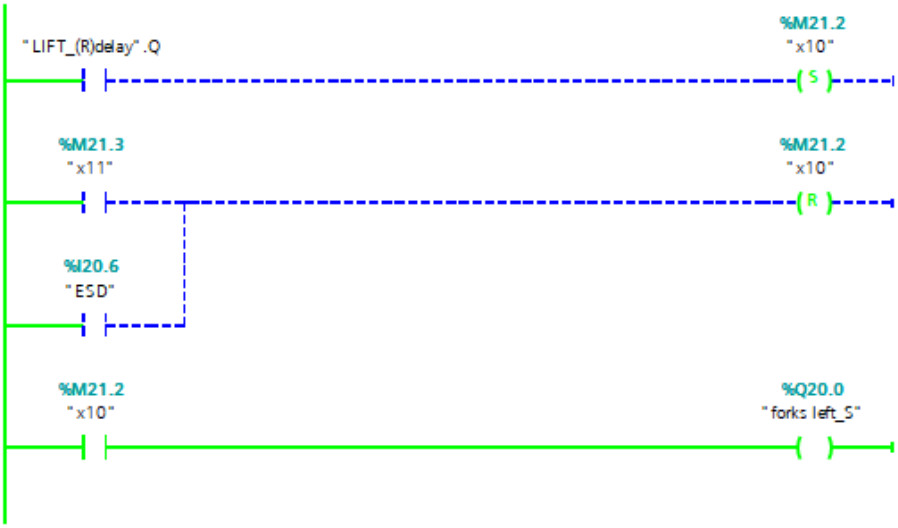



FIGURE 3.17 – Étape 10.

Dans l'étape 11, le capteur "At middle-s" détecte que la fourche a revenu au milieu ce qui va activer l'étape actuelle. La fourche doit revenu à la position de départ avec les mêmes pas. La figure 3.18 représente l'étape 11.Donc ici c'est le rôle des actionneurs "movingX-" et "movingZ-" qui vont déplacer la fourche en arrière durant les mêmes durées : 16s sur l'axe X et avec 9s sur l'axe Z.

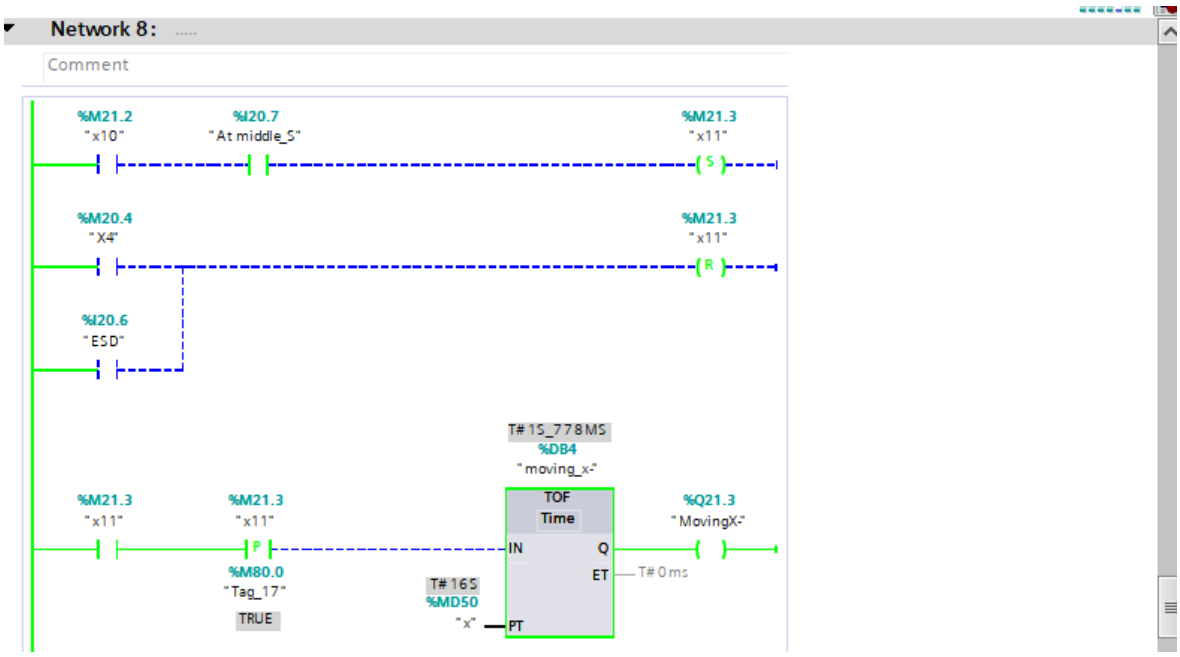



FIGURE 3.18 – Étape 11.

La figure 3.19 représente l'action associée à l'étape 11.

### 3.6.3 Remarques

La dernière branche de l'étape 11 (Network 8) sert à réinitialiser la variable "Stocke" dans le but de recommencer la boucle et faire le stockage d'autres chargements.

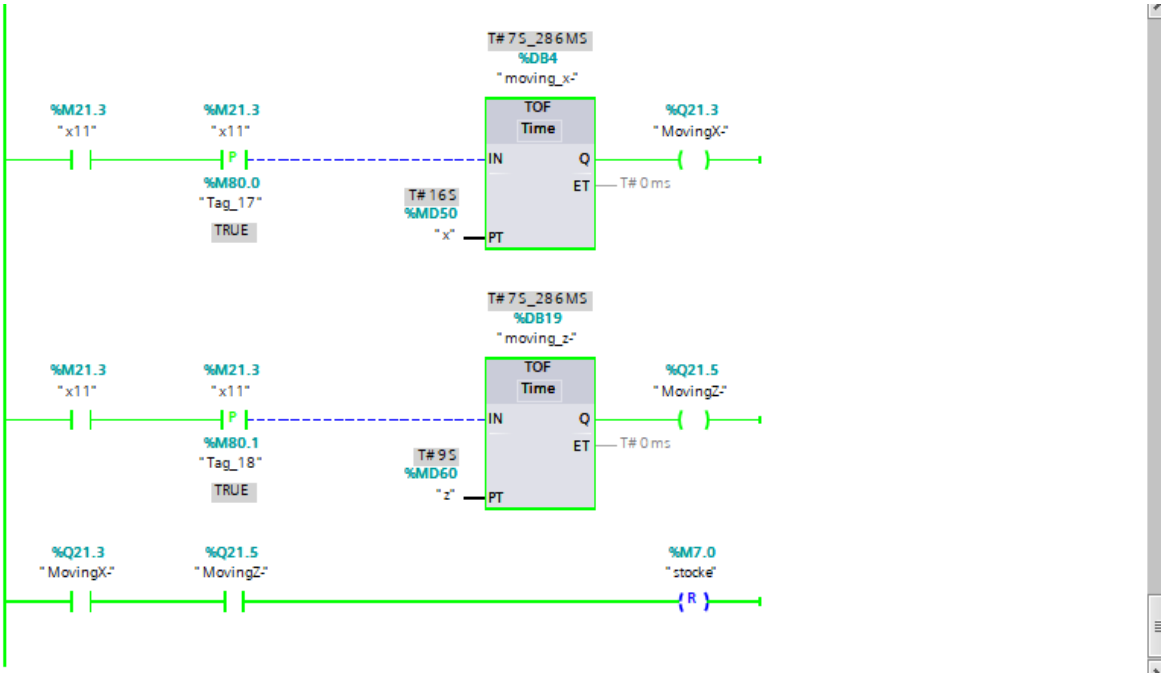



FIGURE 3.19 – movingX- et movingZ-.

### 3.6.4 Déstockage du boîtier

Le Grafcet du déstockage s'agit 8 étapes aussi ( de 30 jusqu'à 37), il est presque identique avec le Grafcet du stockage sauf qu'il y a quelques différences.
Le démarrage du système se fait en étape 30, l'activation de cette étape nous rentre dans le bloc d'emplacement de déstockage, c'est le bloc des données DB8. La figure 3.20 représente l'étape 30 en langage LADDER.
La 3ème branche dans ce réseau a un rôle dans le cas d'un emplacement vide.

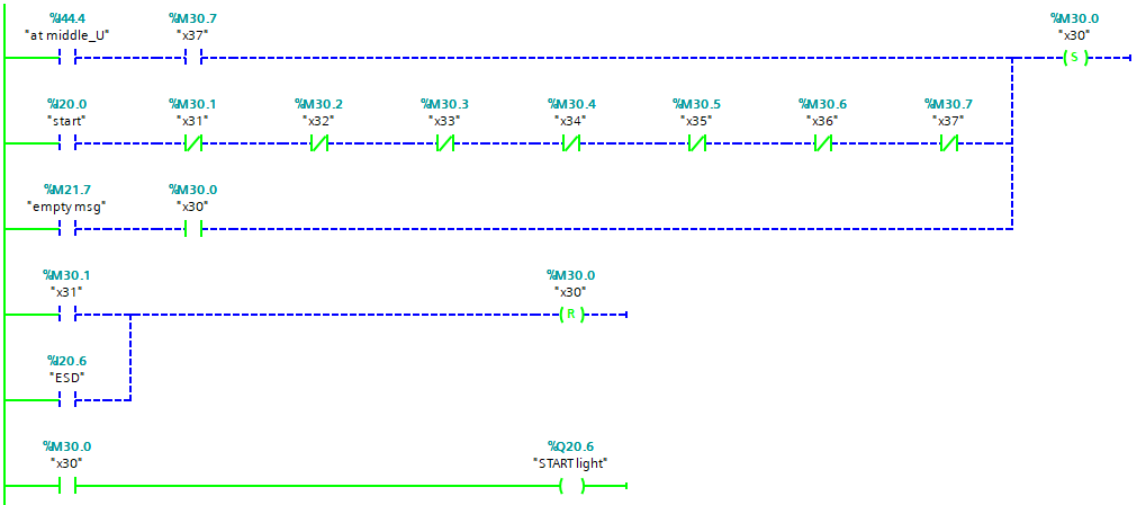



FIGURE 3.20 – Étape 30.

Dans ce cas, on a choisit l'emplacement 16 pour déstocker. Les figures 3.21 et 3.22 représentent le choisit de l'emplacement 16 et son état (occupé) respectivement.

| | | | | | |
|---|---|---|---|---|---|
| "Rooms".Room[3] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[4] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[5] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[6] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[7] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[8] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[9] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[10] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[11] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[12] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[13] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[14] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[15] | %DB1.D... | Bool | FALSE | | FALSE |
| "Rooms".Room[16] | %DB1.D... | Bool | TRUE | | FALSE |

FIGURE 3.21 – Le choisit de l'emplacement 16.

| | | | | | |
|---|---|---|---|---|---|
| "x35" | %M30.5 | Bool | FALSE | FALSE | |
| "empty msg" | %M21.7 | Bool | FALSE | FALSE | |
| "Tag_44" | %MW91 | DEC | 0 | 0 | |
| "consigne unstock" | %MW96 | DEC | 16 | 0 | |
| "x36" | %M30.6 | Bool | FALSE | FALSE | |
| "x37" | %M30.7 | Bool | FALSE | FALSE | |
| "x38" | %M40.0 | Bool | FALSE | FALSE | |
| "x39" | %M40.1 | Bool | FALSE | FALSE | |
| "x40" | %M40.2 | Bool | FALSE | FALSE | |
| "stocke" | %M7.0 | Bool | FALSE | FALSE | |

FIGURE 3.22 – L'emplacement 16 est occupé.

Le bloc des données DB8 contient les mêmes conditions de marche de la fourche, dans notre cas elle se déplace durant 16s sur l'axe X et aussi 16s sur l'axe Z. Les figures 3.23 et 3.24 représente le bloc des données "déstockage" écrit en langage ST (Texte structuré).

```
0001   IF "Rooms".Room["consigne unstock"] = TRUE
0002   THEN
0003     // X moving time
0004     IF "consigne unstock" >= 1 AND "consigne unstock" <= 4
0005     THEN
0006       //"Tag_4" := "consigne stock";
0007       "xx" := T#5s;
0008     ELSIF "consigne unstock" >= 5 AND "consigne unstock" <= 8
0009     THEN
0010       "xx" := T#9s;
0011     ELSIF "consigne unstock" >= 9 AND "consigne unstock" <= 12
0012     THEN
0013       "xx" := T#13s;
0014     ELSIF "consigne unstock" >= 13 AND "consigne unstock" <= 16
0015     THEN
0016       "xx" := T#16s;
0017     END_IF;
```

FIGURE 3.23 – DB8 (partie1).

```
0019     // Z moving time
0020     IF "consigne unstock" = 1 OR "consigne unstock" = 5 OR "consigne
     unstock" = 9 OR "consigne unstock" = 13
0021     THEN
0022       //"Tag_4" := "consigne stock";
0023       "zz" := T#0s;
0024     ELSIF "consigne unstock" = 2 OR "consigne unstock" = 6 OR "con-
     signe unstock" = 10 OR "consigne unstock" = 14
0025     THEN
0026       "zz" := T#9s;
0027     ELSIF "consigne unstock" = 3 OR "consigne unstock" = 7 OR "con-
     signe unstock" = 11 OR "consigne unstock" = 15
0028     THEN
0029       "zz" := T#13s;
0030     ELSIF "consigne unstock" = 4 OR "consigne unstock" = 8 OR "con-
     signe unstock" = 12 OR "consigne unstock" = 16
0031     THEN
0032       "zz" := T#16s;
0033     END_IF;
0034     "unstocke" := TRUE;
0035   ELSE
0036     "empty msg" := TRUE;
0037
0038   END_IF;
```

FIGURE 3.24 – DB8 (partie2).

La figure 3.25 représente le fonctionnement des actionneurs "movingX++" et "movingZ++" qui sont une action associée à l'étape30.

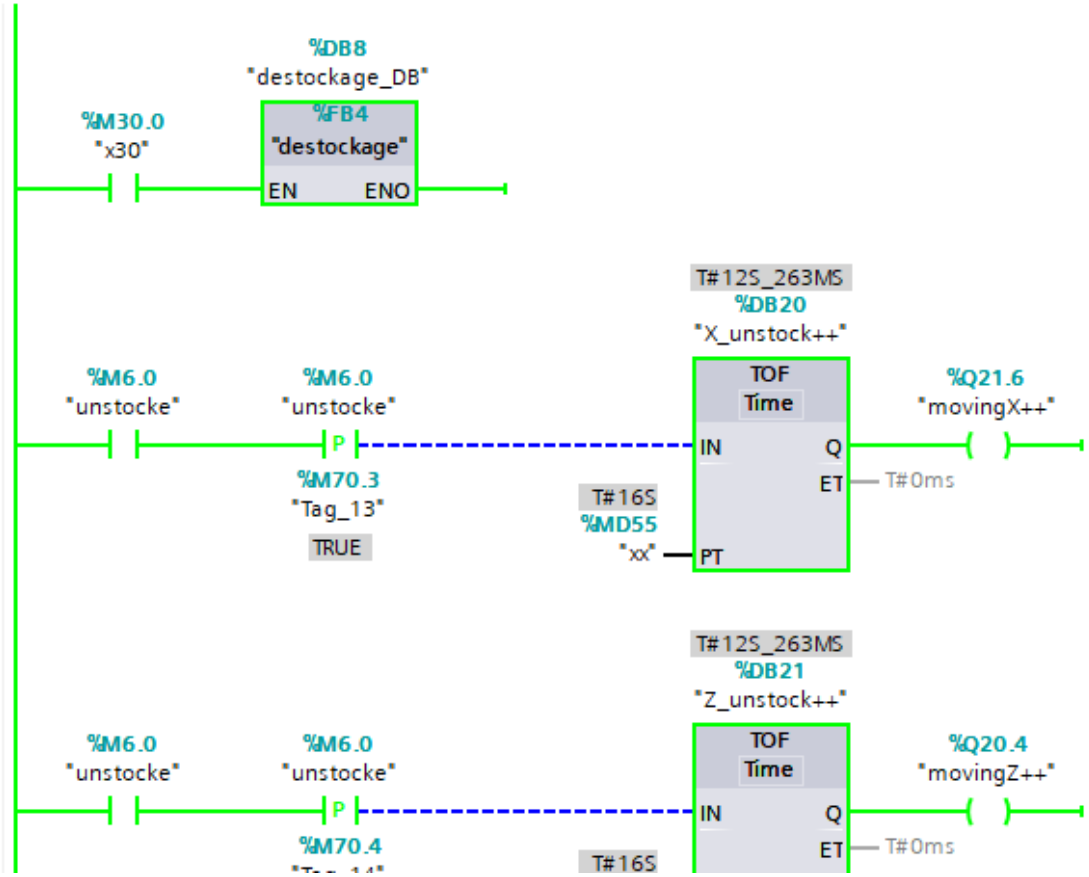



FIGURE 3.25 – movingX++ et movingZ++.

Après on passe à l'étape 31, où la fourche sort vers gauche. La figure 3.26 représente l'étape 31 en langage LADDER.

### 3.6.5 Remarque

L'entrée de chaque temporisateur de type "TOF" est représenté par un contact -|P|- :
car l'activation de ce type de temporisateur se fait par l'entrée d'une impulsion (front montant).

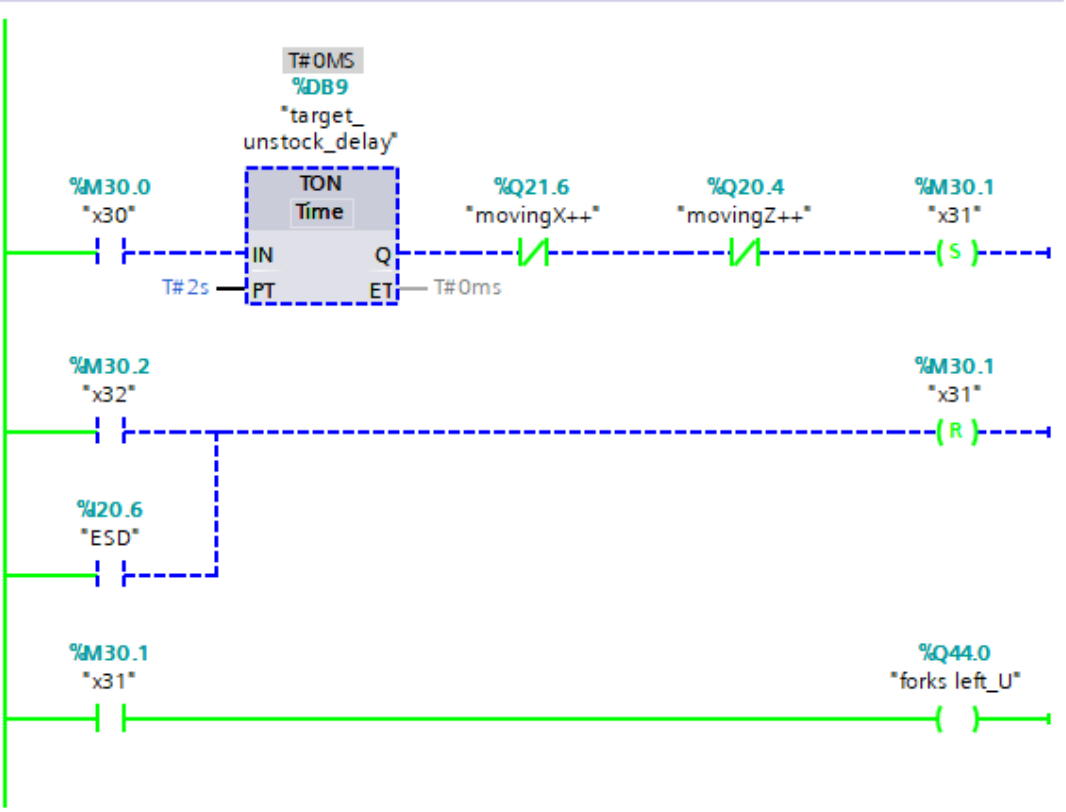



FIGURE 3.26 – Étape 31.

Dans l'étape 32, le capteur "At left-u" détecte la fourche ce qui va provoquer l'élévation du boîtier. La figure 3.27 représente l'étape 32 en langage LADDER.
Dans l'étape 33, la fourche porte la boite au milieu pour retourner au point de dé-

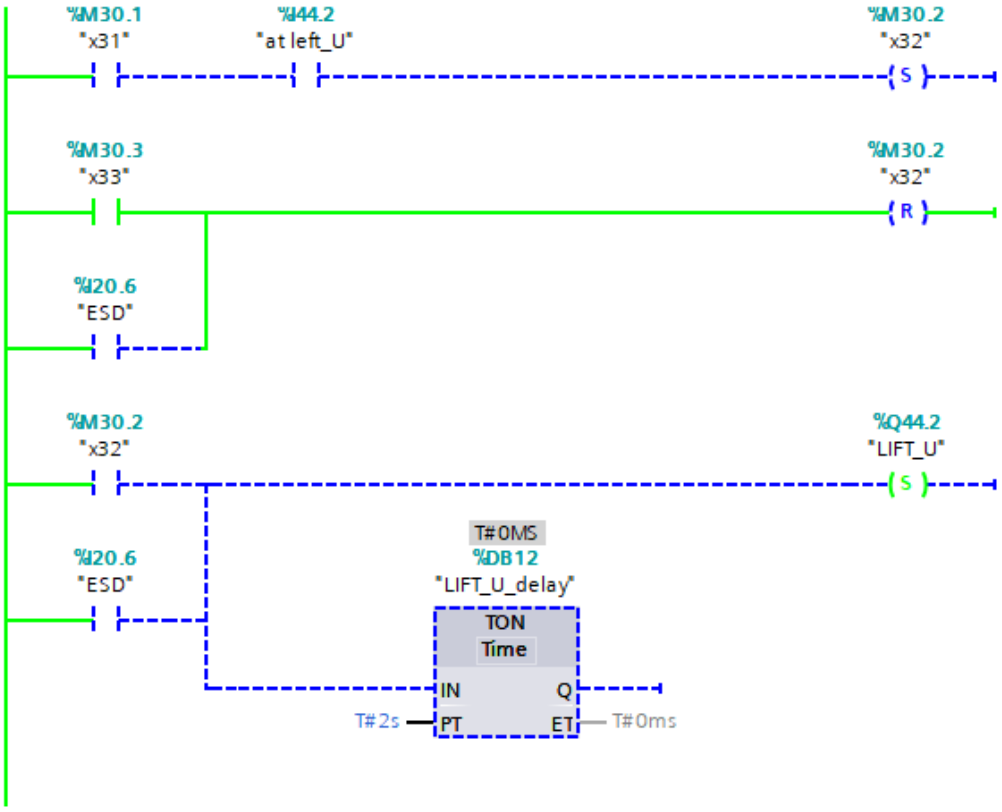



FIGURE 3.27 – Étape 32.

part, donc l'action associée de cette étape est "Forks tight-u". La figure 3.28 représente l'étape 33 en langage LADDER.

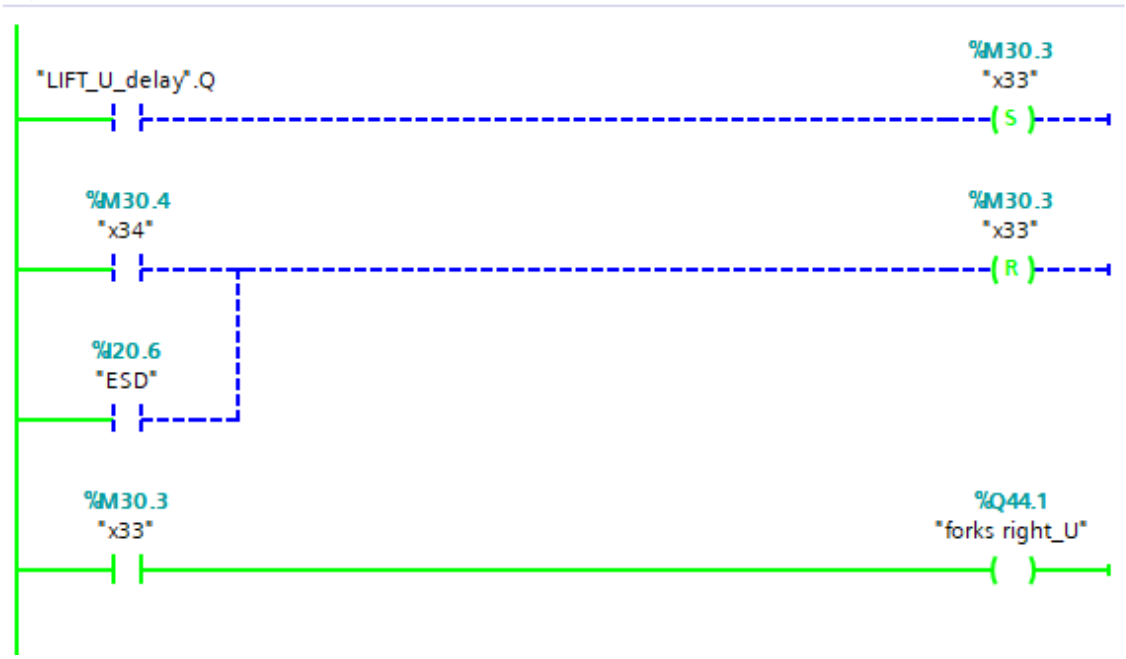



FIGURE 3.28 – Étape 33.

Quand le capteur "At middle-u" détecte la présence de la fourche au milieu, l'étape 34 s'active. L'activation de cette étape marque que l'emplacement 16 est vide d'après le bloc des données DB11. Ce bloc est complètement identique au bloc DB17 qu'on a vu précédemment. La figure 3.29 représente l'étape 34 en langage LADDER.

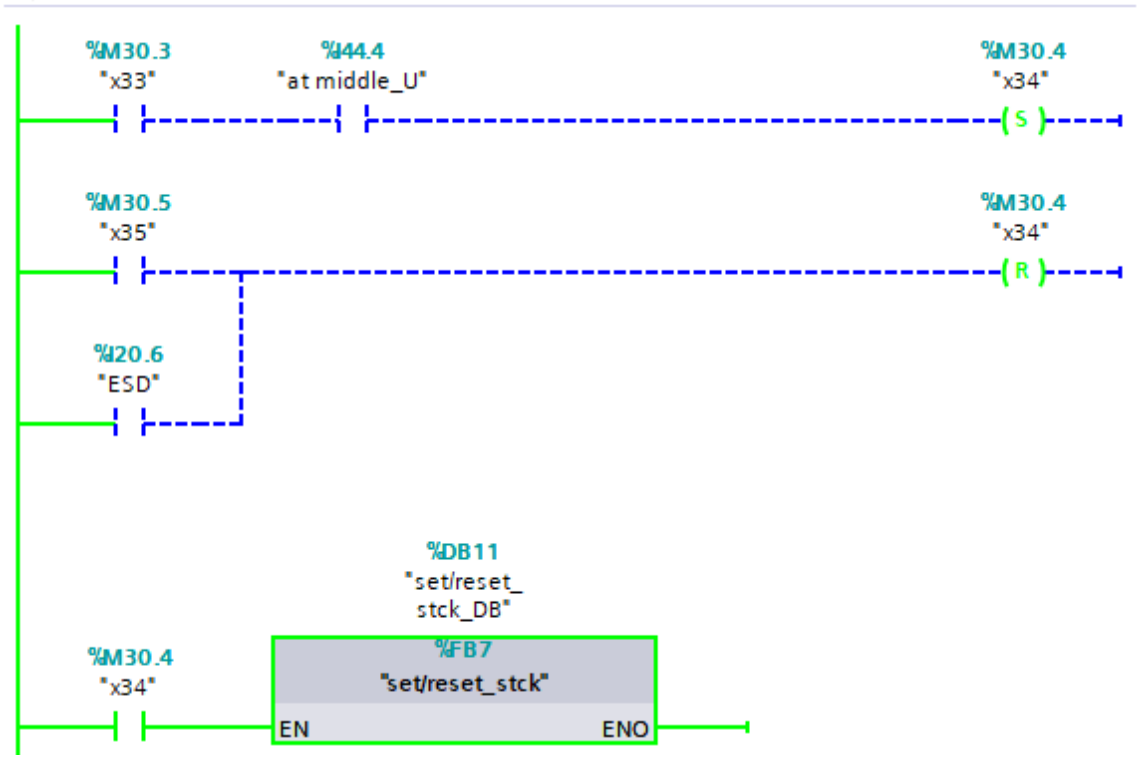



FIGURE 3.29 – Étape 34.

La fourche se déplace au point de départ grâce à les actionneurs "movingX–" et movingZ–" durant 16s sur les deux axes X et Z, ce qui bien décrit dans la figure 3.30.

Les étapes 35, 36 et 37 sont représentées dans les figures 3.31 , 3.32 et 3.33 respectivement. Dans ces 3 étapes , la fourche mets la boite sur le convoyeur de sortie pour le décharger.

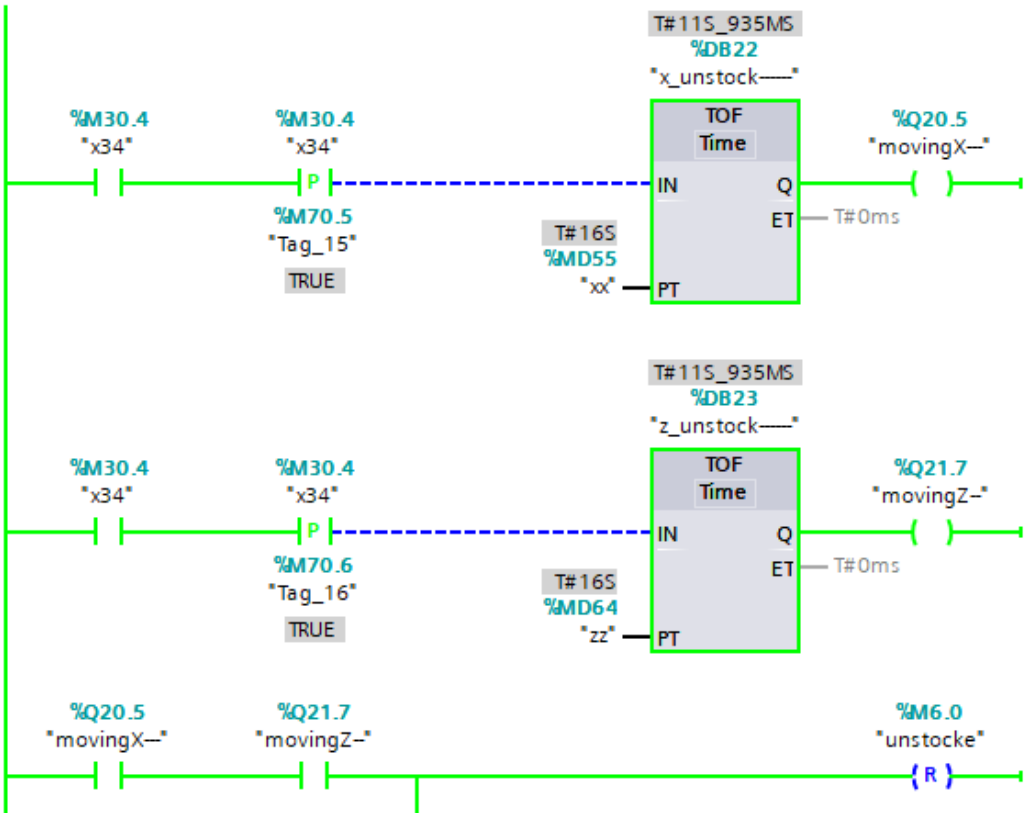



FIGURE 3.30 – movingX– et movingZ–.

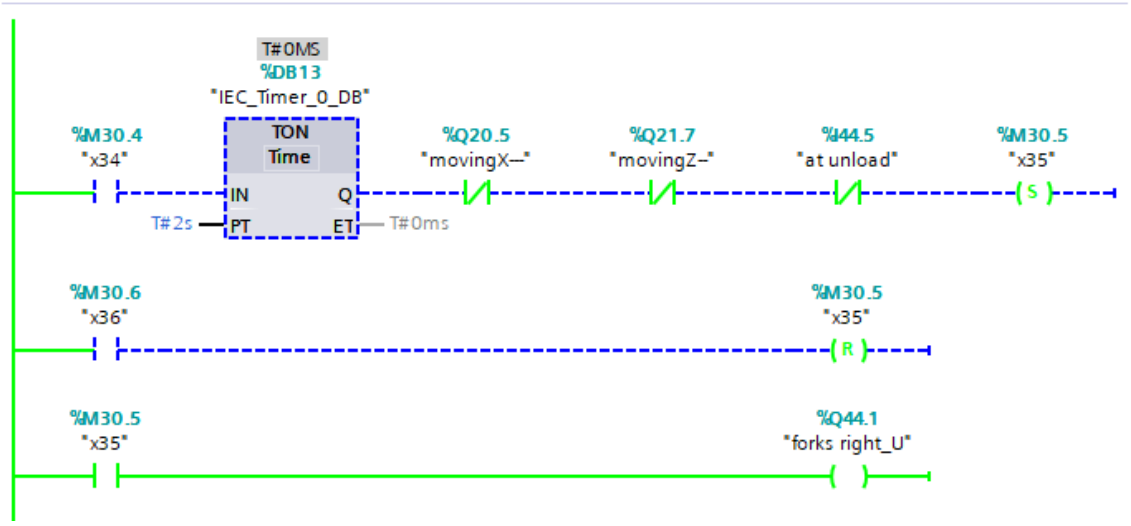



FIGURE 3.31 – Étape 35.

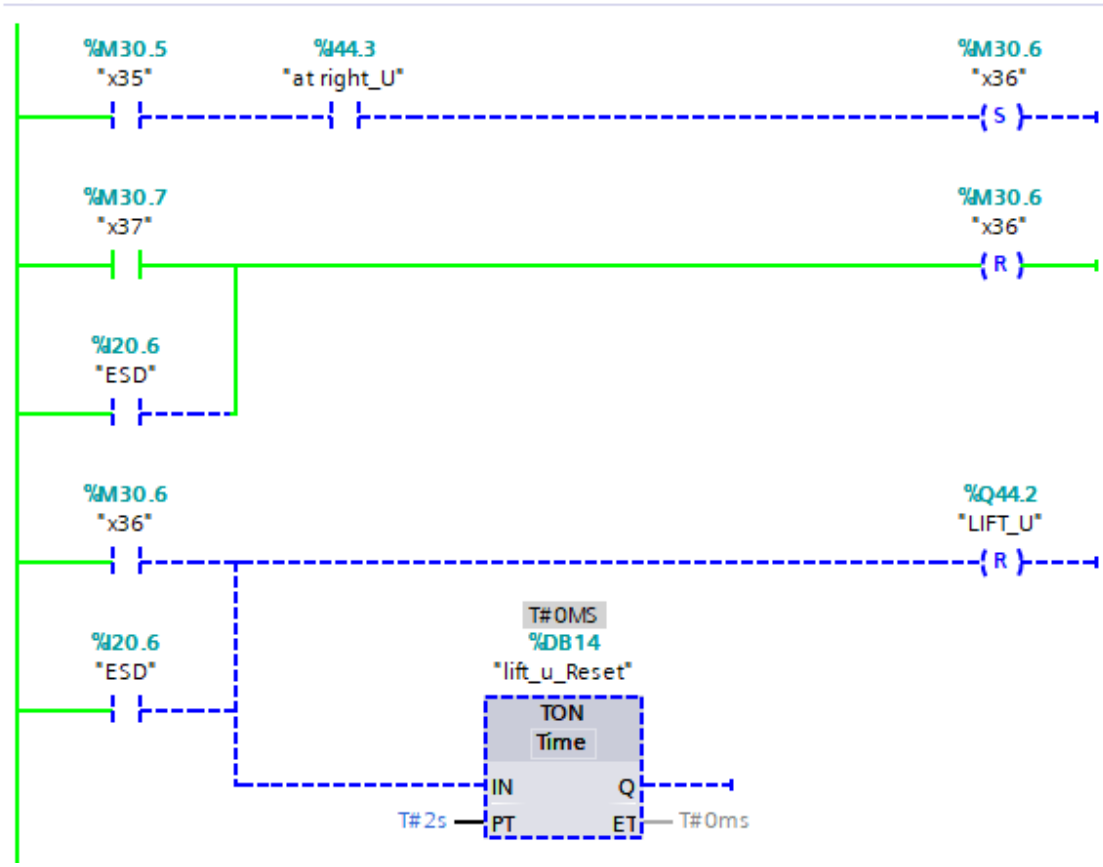



FIGURE 3.32 – Étape 36.

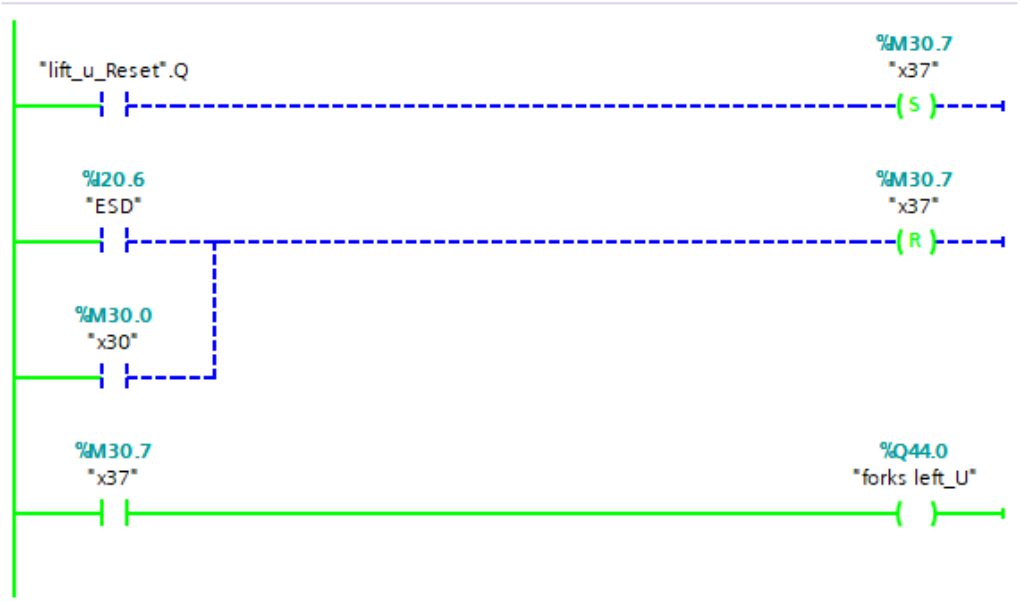



FIGURE 3.33 – Étape 37.

### 3.6.6 Convoyeur de sortie

Le convoyeur de sortie fait le déchargement de la pièce, il s'agit de 3 étapes (de 38 jusqu'à 40).
Quand le capteur "At unload" détecte la présence du boîtier sur le convoyeur, ce dernier se démarre pour faire déplacer le boîtier jusqu'au le capteur "At exit" de fin de course et faire l'évacuation de la boite par l'actionneur "Remover". Les figures 3.34, 3.35 et 3.36 représentent les étapes 38, 39 et 40 respectivement en langage LADDER.

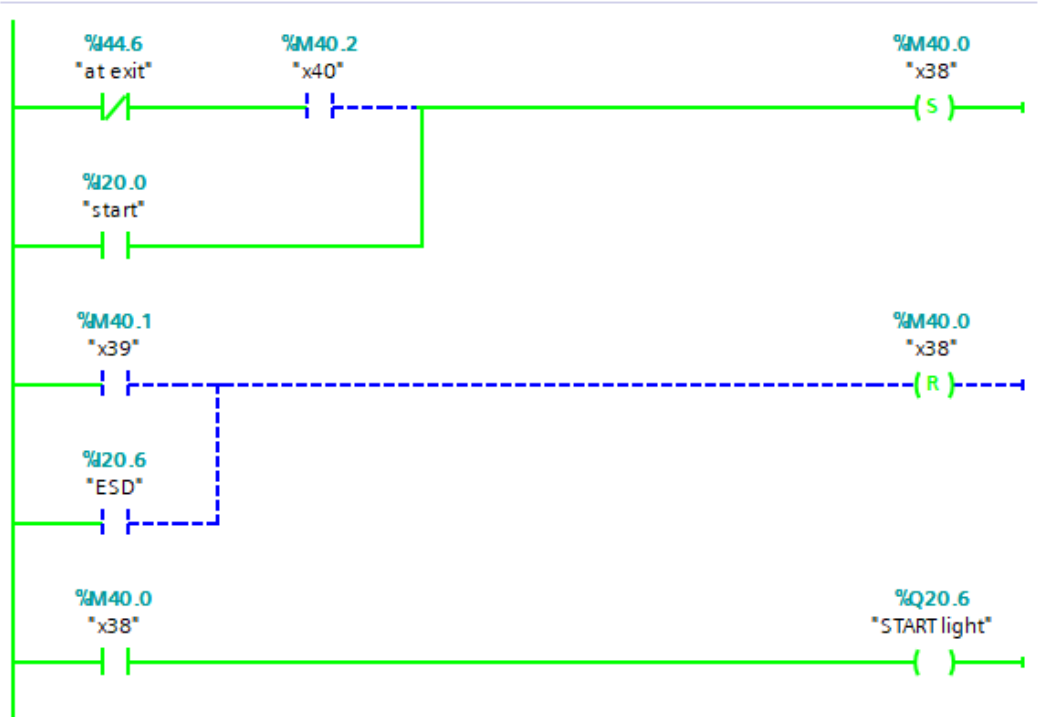



FIGURE 3.34 – Étape 38.

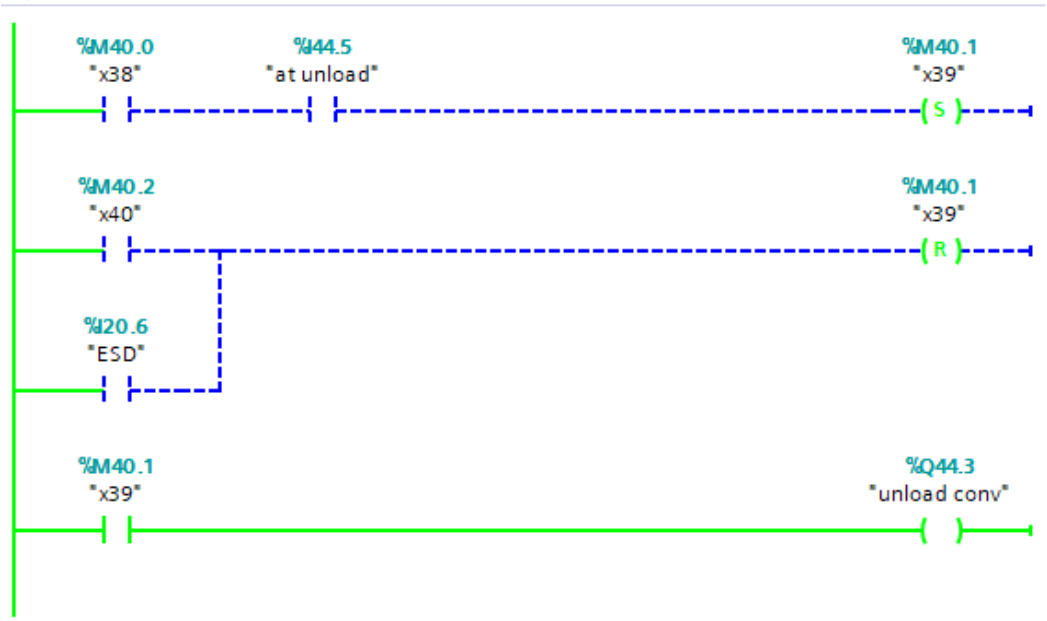



FIGURE 3.35 – Étape 39.

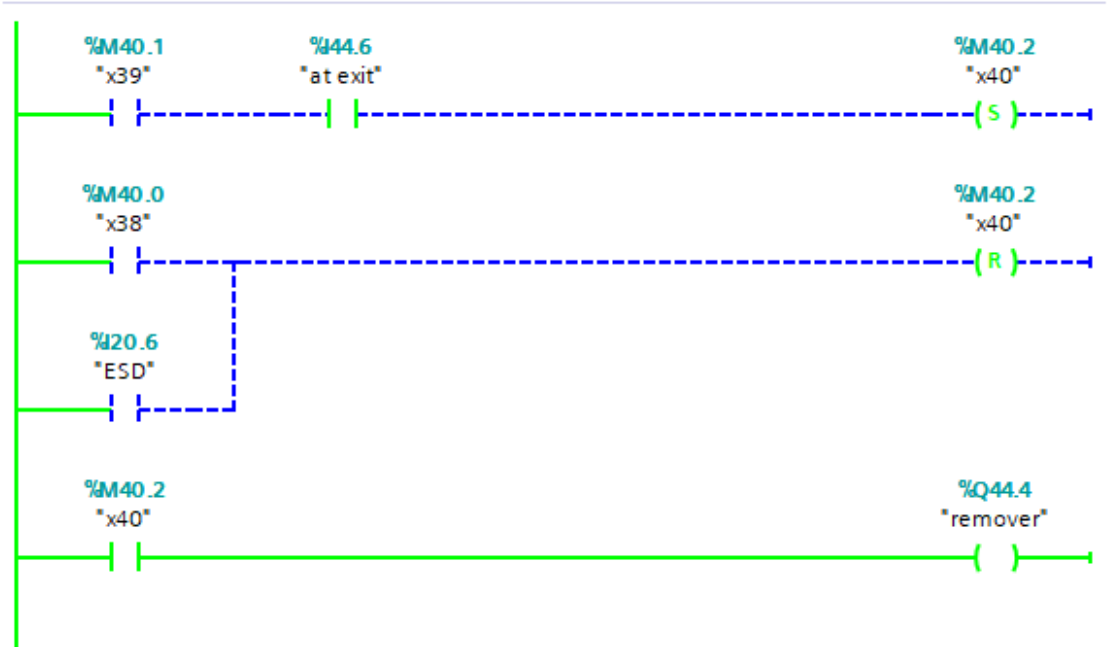



FIGURE 3.36 – Étape 40.

## 3.7 Création de dispositif IHM

Dans notre projet on a introduit un nouvel objet qui est le dispositif IHM en choisissant le type de pupitre sur lequel, pour notre application on a utilisé un HMI [TP1200 Comfort] touche ecran 12.1" TFT display, 1280 x 800 pixels, 16M colors ; Touch screen ; 1 x MPI/PROFIBUS DP, 1 x PROFINET/Industrial Ethernet interface with MRP and RT/IRT support (2 Ports) ; 2 x Multimedia card slot ; 3 x USB.
Il faut établir une liaison entre l'IHM et la station API pour permettre de lire les données qui se trouvent dans l'automate. La liaison est établie en choisissant le protocole de communication Ethernet 3.37.

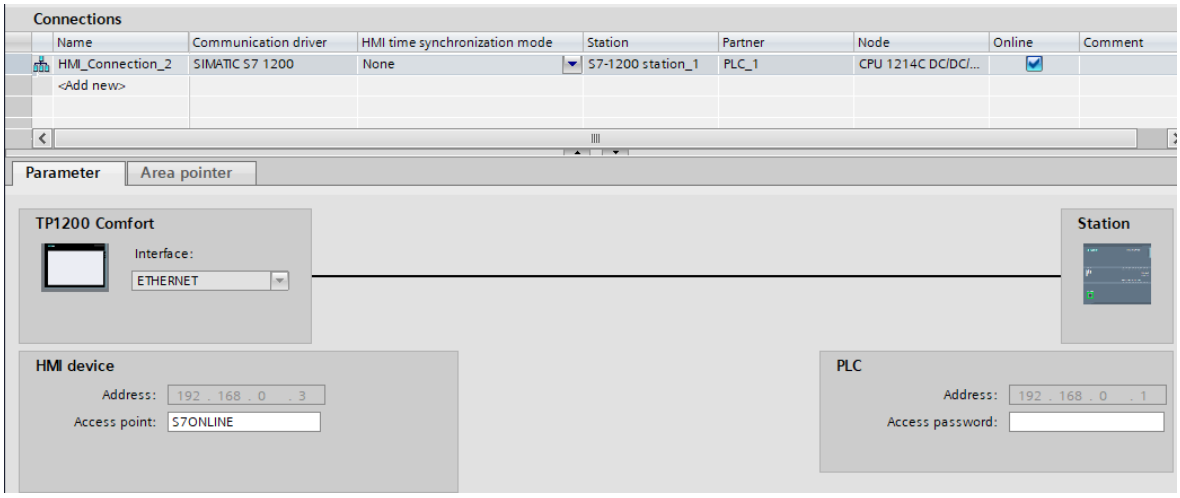



FIGURE 3.37 – Liaison entre HMI et API.

## 3.8 Les vues de supervision

Chaque vue représente de manière synoptique animée le schéma de principe de fonctionnement de l'installation .

### 3.8.1 La vue du menu principal

C'est une vue globale qui permet le passage d'une vue à une autre 3.38. Elle contient les boutons "Start" et "ESD" , ainsi que les champs d'entrée de l'emplacement désiré.

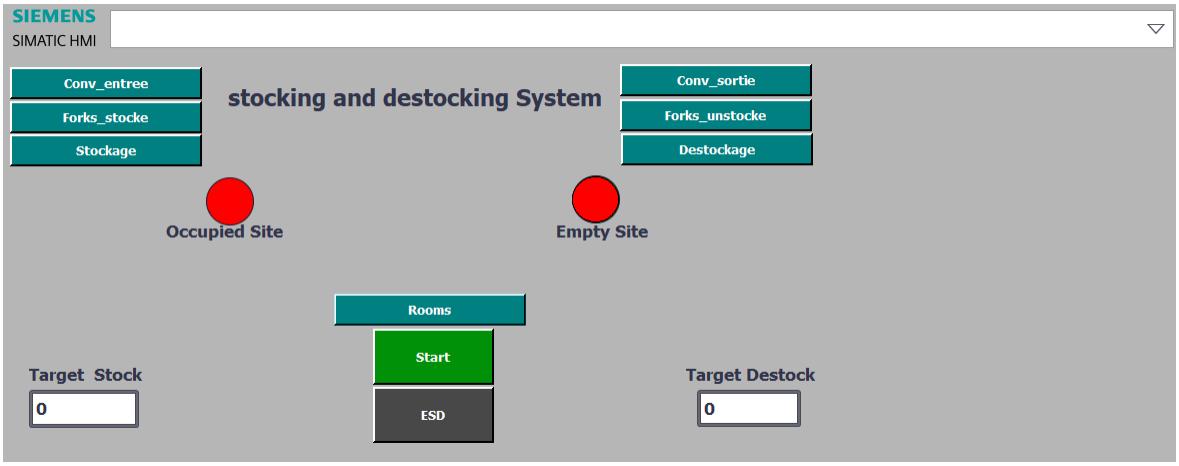



FIGURE 3.38 – La vue principale.

### 3.8.2 La vue du convoyeur d'entrée

C'est une vue qui permet la surveillance du fonctionnement du convoyeur d'entrée 3.39. Dans ce cas le capteur "At entry" est dans un état de détection.

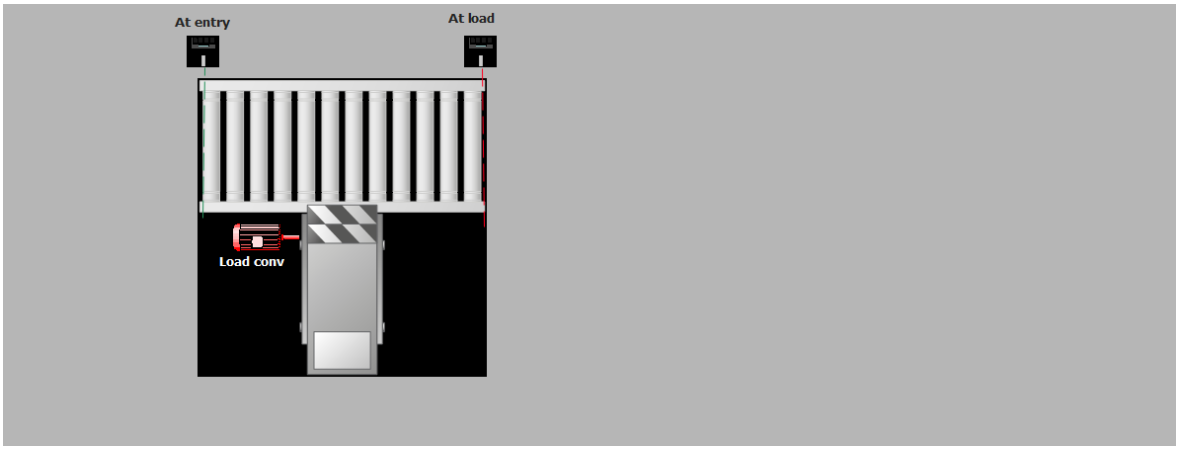



FIGURE 3.39 – La vue du convoyeur d'entrée.

### 3.8.3 La vue du convoyeur de sortie

C'est une vue qui permet la surveillance du fonctionnement du convoyeur de sortie 3.40.

### 3.8.4 La vue de stockage/déstockage

C'est une vue qui permet la surveillance du fonctionnement du système stockage et déstockage. La figure 3.41 représente la fonctionnement du système stockage et puisque les deux systèmes sont identique en terme de schéma synoptique donc on a représenté

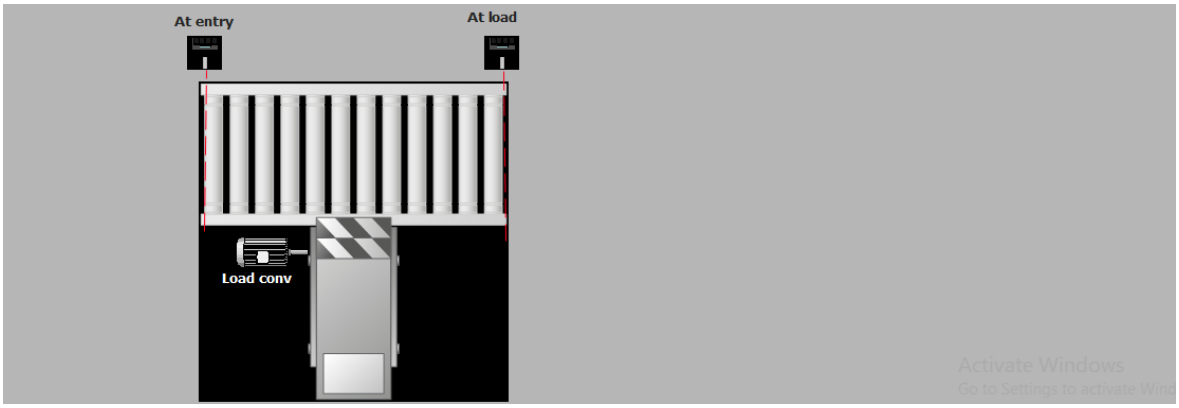



FIGURE 3.40 – La vue du convoyeur de sortie.

l'une des deux.

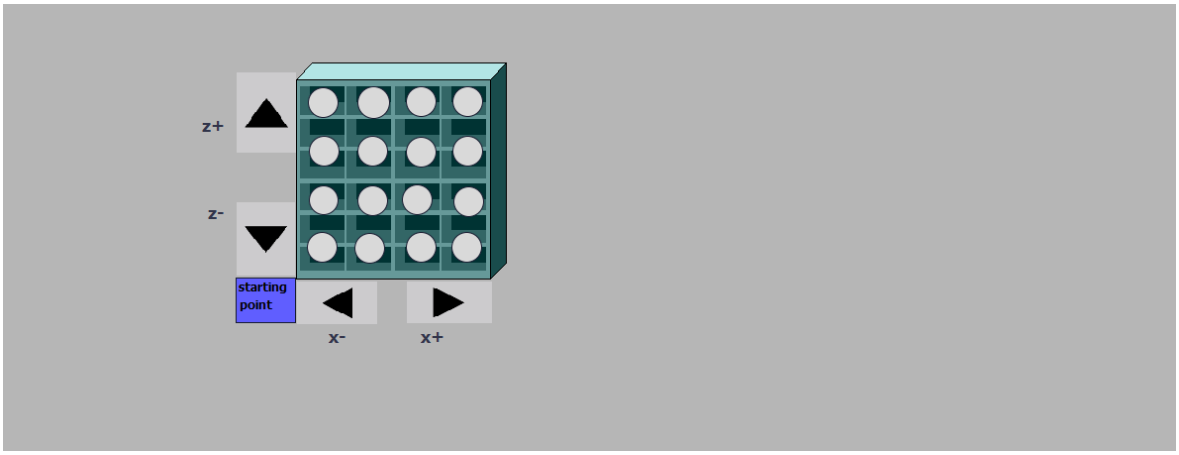



FIGURE 3.41 – La vue du système stockage.

### 3.8.5 La vue des emplacements

C'est une vue qui permet la surveillance des états des emplacements 3.42. Dans ce cas l'emplacement 16 est occupé.

### 3.8.6 La vue de la fourche

C'est une vue qui permet la surveillance les états des capteurs "At left" , "At middle" et "At right" qui accompagnent la fourche 3.43. Dans ce cas le capteur "At middle" est dans un état de détection.

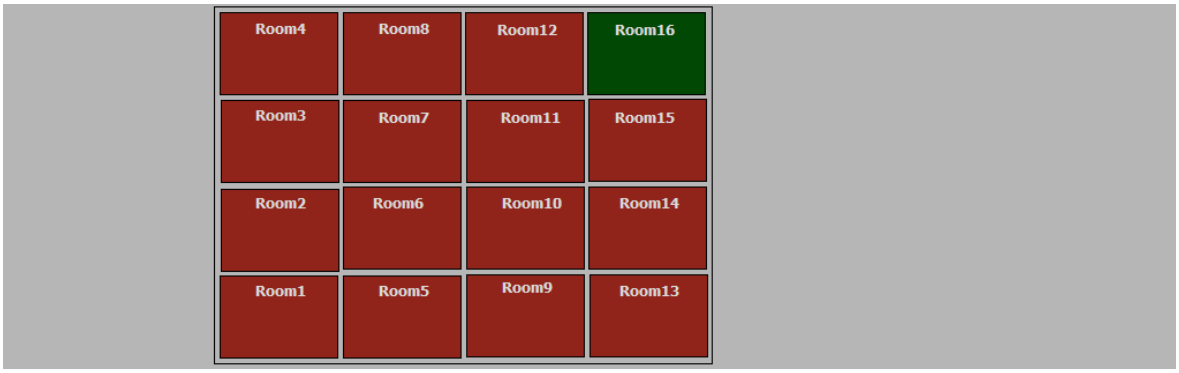



FIGURE 3.42 – La vue des emplacements.

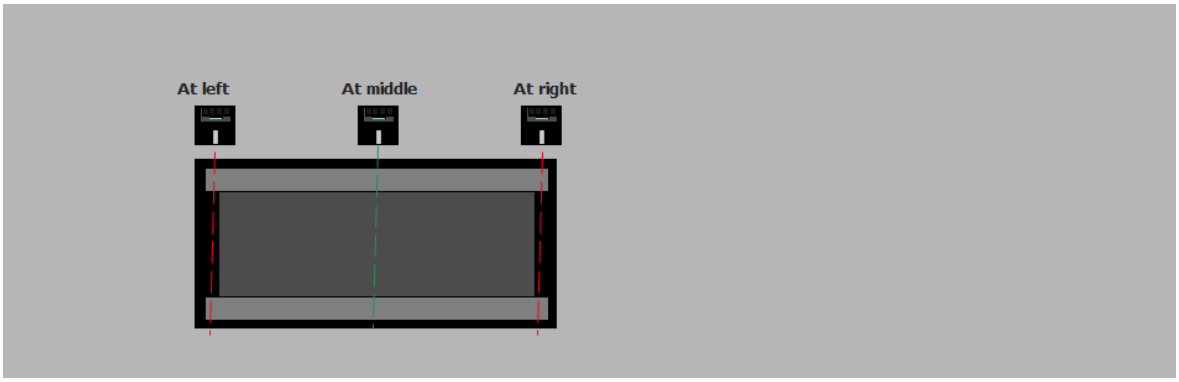



FIGURE 3.43 – La vue de la fourche.

## 3.9 Conclusion

Dans ce chapitre nous avons présenté en premier lieu les différents logiciels dans l'environnement TIA Portal ainsi que le WINCC sous le même environnement qui a fourni une grande aide à faciliter la tâche de supervison du programme.
Ensuite nous avons présenté notre réalisation pratique en introduisant les étapes du Grafcet en langage LADDER.

# Conclusion générale

A l'issue de notre travail, nous pouvons conclure que :
Pour être à la page avec les exigences introduites par l'évolution des industries, la commande des processus avec un automate programmable industriel est la solution souvent recherchée, vue la justesse des traitements numériques que les APIs effectuent pour générer la commande adéquate à tout moment et dans toutes les conditions.
L'utilisation de l'environnement TIA Portal, qui est le dernier logiciel d'ingénierie de Siemens nous a permis d'atteindre notre objectif de projet. Afin de réaliser notre système, la représentation du diagramme fonctionnel Grafcet nous a autorisé d'entreprendre l'étude du système à évolution séquentielle, ainsi que la programmation grâce à le STEP 7 et la surveillance du procédé en utilisant la supervision par le WinCC.
Cependant, la réalisation d'un bon système à simuler nécessite la connaissance de certains concepts intègres dans les nouvelles technologies informatiques, consistant au contrôleur logique programmable, ses caractéristiques et ses propriétés, ainsi que les langages de programmation possibles et utilisés. Dans notre travail, on a réussi à écrire le cahier de charge du système stockage/déstockage, sa description par le Grafcet et de faire la programmation de ce système en langage à contacts.
En dernière analyse, on tient en conséquence à décrire qu'une bonne automatisation d'un procédé doit être performante et d'un coût optimal. Cela sera obtenu en passant par :

- L'élaboration d'un cahier des charges qui comprend tous les aspects fonctionnels du processus.
- La modélisation du cahier des charges par un des outils de modélisation comme le Grafcet.
- Le choix optimal de l'API, les actionneurs et les capteurs.

On veux bien mentionné que notre premier objectif c'était de réaliser ce système du Stockage/Déstockage sur le terrain en utilisant la maquette disponible au niveau de laboratoire du constructeur LUCAS-NÜLLE, mais malheureusement à cause de plusieurs facteurs on n'a pas pu de le faire. Parmi ces facteurs : l'impossibilité d'établir une connexion entre l'API et la maquette.

En raison du manque du temps, on prévoit d'élaborer les points suivants :

— Un système de trie par taille qui consiste à mettre les pièces les plus grandes dans des étages à base altitude.
— Protocole de communication sans fil par wifi (l'industrie 4.0) .
— Installer des capteurs de position dans chaque emplacement de l'armoire pour assurer une meilleure fiabilité et précision mais elle reste coûteuse.

Nous espérons enfin que notre projet peut être une phase éducative pour les travaux pratiques des systèmes automatisés à notre département et rendre service à tous ceux qui aborderont le même sujet et obtient la satisfaction de nos encadreurs et les membres du jury.

# Bibliographie


[1] Fahem Hammar. Etude de l'automatisation par automate programmable S7-300 de la machine à garnir les encoches de l'ENEL. https://www.ummto.dz/dspace/bitstream/handle/ummto/9085/FAHEMNASSIM_HAMMARYAZID.pdf?sequence=1.

[2] harmonicdrive. La technique de l'automatisation. https://harmonicdrive.de/fr/glossaire/la-technique-de-lautomatisation.

[3] I IA AS S ID 1. S7-PLCSIM V5.4 . https://cache.industry.siemens.com/dl/files/828/54667828/att_110512/v1/s7wsvhdc_fr-FR.pdf.

[4] L. BERGOUGNOUX (POLYTECH' Marseille 2004–2005). Cours grafcet : Les notions de base . https://www.technologuepro.com/cours-automate-programmable-industriel/Cours-Grafcet-notions-de-base.htm.

[5] motorline. 5 Raisons d'investir dans l'automatisation industrielle. https://motorline.pt/fr/blog/5-raisons-dinvestir-dans-lautomatisation-industrielle/.

[6] Mr L. BERGOUGNOUX. Les Automates Programmables Industriels (API). https://www.technologuepro.com/cours-automate-programmable-industriel/Les-automates-programmables-industriels-API.htm.

[7] pro-face. Qu'est-ce qu'une interface Homme-Machine ou IHM ? https://www.proface.com/fr/node/49508.

[8] SchoolMouv. Systemes automatises. https://www.schoolmouv.fr/cours/systemes-automatises/fiche-de-cours.

[9] Vidya Muthukrishnan. programmable-logic-controllers/Structured-Text-Programming. https://www.electrical4u.com/programmable-logic-controllers/#Structured-Text-Programming.

[10] Vladimir Romanov. plc-programming-languages . https://www.solisplc.com/blog/plc-programming-languages.

[11] Alain GONZAGA. cours-automate-programmable-industriel/Les-automates-programmables-industriels-API. https://www.technologuepro.com/cours-automate-programmable-industriel/Les-automates-programmables-industriels-API.htm.

[12] automation-sense. Cours/Formation Wincc Flexible Siemens . https://www.automation-sense.com/blog/automatisme/cours-formation-wincc-flexible-siemens.html.

[13] Automatisation et technique des commandes - SCE . Document de formation T I A . https://www.automation.siemens.com/sce-static/learning-training-documents/classic/basics-programming/a03-startup-fr.pdf.

[14] bennisnajib. cours sur le grafcet . https://www.specialautom.net/automatisme/Grafcet.pdf.

[15] chintglobal. programmable-logic-controller-plc. https://chintglobal.com/blog/programmable-logic-controller-plc/.

[16] copadata. quest-ce-qu-une-ihm-interface-homme-machine. https://www.copadata.com/fr/produits/zenon-software-platform/visualisation-controle/quest-ce-qu-une-ihm-interface-homme-machine-copa-data/.

[17] documents. Introduction-Au-Logiciel-TIA-Portal . https://fr.scribd.com/document/256910981/2-Introduction-Au-Logiciel-TIA-Portal.

[18] e Soraya Damasio Bertoncello. Qu'est-ce que l'automatisation industrielle . https://www.novus.com.br/blog/quest-ce-que-lautomatisation-industrielle/?lang=fr.

[19] espacetechnologue. upport-de-formation$_W CC - Flexible - Runtime$..