المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

SIEMENS

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique

Rapport de Stage

TIA-MICRO2 / Maintenace 1

Réalisé par : BELDJOUD Oubaid ellah
Spécialité : Automatique
Entreprise d'accueil : SIEMENS
Encadré par : Pr.MOUFID Mansour / Madame BELARIF Nesrine

Période du stage : Du 14 au 25 décembre 2025

Année universitaire : 2025 – 2026

Remerciements

Je tiens à exprimer mes sincères remerciements à toutes les personnes qui ont contribué, de près ou de loin, au bon déroulement de mon stage et à la réussite de cette formation.

Je remercie tout particulièrement **Monsieur MOUFID Mansour**, mon tuteur de la formation **TIA-MICRO2**, pour son encadrement, sa pédagogie et sa disponibilité tout au long des cinq jours de formation.

J'adresse également mes remerciements à **Madame BELARIF Nesrine**, mon encadreure durant la formation **Maintenance 1**, pour ses explications claires, ses conseils précieux et son accompagnement durant cette étape importante de mon apprentissage.

Je tiens à remercier **Monsieur BRADAI Abderrahmane** pour son orientation et son aide, ainsi que pour m'avoir guidé vers mes encadreurs et facilité mon intégration au sein de l'environnement de formation.

Mes remerciements s'adressent aussi à l'ensemble des **ingénieurs de Sonatrach et de Naftal** qui ont suivi la première formation avec moi et qui m'ont aidé tout au long de la formation par leurs conseils et leur professionnalisme.

Je veux aussi remercier **Monsieur LAMRI Amine** pour sa présentation qui contenait des notions précieuses sur Mobility (Metro).

j'adresse mes remerciements à **Madame DAMECHE Souad** pour m'avoir facilité les démarches administratives nécessaires au bon déroulement du stage.

Enfin, je remercie profondément **mon père** pour son soutien constant, ainsi que **Monsieur Malik**, pour son encouragement et son appui moral tout au long de mon parcours.

Résumé

Ce rapport présente les travaux réalisés lors d'un stage de formation technique au sein de Siemens, s'étant déroulé sur deux périodes distinctes de cinq jours chacune. La première formation, TIA-MICRO2, a porté sur la programmation avancée d'automates SIMATIC S7-1200 avec le logiciel TIA Portal, couvrant des aspects tels que le traitement des signaux analogiques, la gestion des blocs de données, la communication industrielle (ISO-on-TCP, PROFINET), la configuration d'IHM, l'utilisation d'objets technologiques (PID, Axis) et l'introduction au langage SCL. La seconde formation, Maintenance 1, a été consacrée à la prise en main des automates S7-300 avec le logiciel STEP7, incluant la configuration matérielle, la programmation en langage Ladder et le diagnostic de panne.

Ce stage a permis l'acquisition de compétences pratiques dans la mise en service, la programmation et la maintenance de systèmes automatisés, tout en offrant une vision concrète des solutions industrielles de Siemens. Les différentes manipulations réalisées sur bancs d'essai et maquettes pédagogiques ont consolidé les connaissances théoriques et développé une méthodologie de travail adaptée aux environnements industriels.

Table des matières

Table des figures

Chapitre 1

Introduction

L'automatisation industrielle représente aujourd'hui un pilier essentiel de la compétitivité des entreprises, permettant d'optimiser les processus de production, d'améliorer la qualité et d'assurer une meilleure fiabilité des installations. Dans ce contexte, la maîtrise des outils de programmation et de configuration des automates programmables industriels (API) devient une compétence cruciale pour les ingénieurs en automatisme.

Ce stage, organisé par Siemens, leader mondial dans le domaine de l'automatisation, avait pour objectif de nous familiariser avec deux environnements majeurs de la gamme SIMATIC : les automates S7-1200 programmés sous TIA Portal et les S7-300 programmés sous STEP7. À travers deux formations intensives - TIA-MICRO2 et Maintenance 1 - nous avons pu aborder un large spectre de fonctionnalités, depuis la configuration de base jusqu'à des aspects avancés comme la communication réseau et les objets technologiques.

Ce rapport décrit dans un premier temps le contexte de ce stage au sein de Siemens, puis présente de manière détaillée les différentes activités réalisées pendant les deux formations. Il analyse ensuite les apports de cette expérience sur le plan technique et professionnel, avant de conclure sur les perspectives ouvertes par cette immersion dans l'écosystème industriel Siemens.

Chapitre 2

Présentation de l'entreprise

2.1 Historique et activités

Fondée en 1847 par Werner von Siemens, Siemens AG est un conglomérat industriel allemand mondialement reconnu, présent dans plus de 190 pays. Initialement spécialisée dans la télégraphie, l'entreprise a diversifié ses activités au fil des décennies pour couvrir aujourd'hui plusieurs domaines majeurs : automatisation industrielle, technologies médicales, infrastructures énergétiques, mobilité intelligente et bâtiments techniques.

La division Digital Industries (DI), au sein de laquelle s'inscrit notre stage, se consacre aux solutions d'automatisation et de digitalisation pour l'industrie. Elle propose une gamme complète de produits et services incluant les automates programmables SIMATIC, les systèmes de commande, les logiciels d'ingénierie (TIA Portal, STEP7), les technologies de mouvement et les solutions d'IIoT (Industrial Internet of Things).

2.2 Organisation et service d'accueil

Le stage s'est déroulé au sein de l'entreprise Siemens SPA à Hydra. L'encadrement était assuré par des formateurs experts certifiés Siemens, disposant d'une solide expérience terrain. L'environnement de travail mettait à disposition des équipements pédagogiques de dernière génération, reproduisant fidèlement les conditions réelles d'exploitation industrielle.

2.3 Moyens matériels

Le centre de formation dispose d'équipements modernes et variés :

— Automates SIMATIC S7-1200 et S7-300 avec leurs modules d'extension

— Interfaces Homme-Machine (IHM) tactile SIMATIC

— Stations d'E/S distribuées ET200S

— Maquettes pédagogiques de convoyeurs et systèmes mécatroniques

— Bancs de régulation PID et contrôle de moteurs pas-à-pas

— Postes de programmation équipés des logiciels TIA Portal.

Cet environnement technique complet a permis une approche pratique immédiate, alternant théorie et manipulations concrètes sur équipements réels.

Chapitre 3

Description du stage

3.1 Durée et déroulement

5 jours de formation TIA MICRO2 du 14 au 18/12/2025 puis 5 jours sur Maintenace 1 du 21 au 25/12/2025.

3.2 Objectifs pédagogiques

Les objectifs pédagogiques de ce stage étaient clairement définis et structurés selon les deux formations dispensées :

Pour la formation TIA-MICRO2 :

- Maîtriser l'environnement de développement TIA Portal (configuration matérielle, gestion des tags, compilation, téléchargement)
- Implémenter des traitements avancés de signaux analogiques (conversion, normalisation, validation)
- Gérer les blocs de données et structures complexes (DB, STRUCT, ARRAY)
- Mettre en œuvre des communications industrielles (ISO-on-TCP entre CPU, configuration PROFINET)
- Configurer des interfaces Homme-Machine (écrans tactiles) avec gestion d'alarmes et synchronisation
- Utiliser les objets technologiques (PID pour la régulation, Axis pour le contrôle de mouvement)
- Découvrir le langage de programmation SCL pour les algorithmes complexes
- Développer une méthodologie de diagnostic et correction d'erreurs

Pour la formation Maintenance 1 :

- Prendre en main l'environnement STEP7 pour automates S7-300

— Configurer la structure matérielle d'un rack S7-300 (modules CPU, E/S, communication)

— Programmer en langage Ladder (LAD) avec gestion des mémoires et séquences

— Réaliser des séquences automatisées avec temporisations et comptages

— Diagnostiquer et résoudre des dysfonctionnements logiques

— Effectuer des transferts de programmes (upload/download) et sauvegardes

Ces objectifs visaient à fournir une compréhension complète du cycle de vie d'un projet d'automatisation, de la conception à la maintenance.

Chapitre 4

Formation TIA-MICRO2

4.1 Matériel de formation avec S7-1200

(a) SIMATIC Field PG

(b) Mallete de formation avec S7-1214, touchpanel et simulateur (PLC1)

(c) Cas de formation avec S7-1211, moteur pas à pas une boucle de commande (PLC2)

(d) Configuration de la formation E / S distribuée ET 200S

(e) Modéle de convoyeur

FIGURE 4.1 – Matériels utilisés pendant la formation

FIGURE 4.2 – PLC1 — CPU et modules d'entrées/sorties

Description du PLC1 : L'ensemble est constitué de plusieurs composants essentiels. Le *PM* (Power Module) assure l'alimentation électrique de l'automate et des modules associés. Le *CSM* (Communication/Connection System Module) permet le câblage et la liaison du PLC1 avec les autres composants du système.

Le cœur du système est la *CPU 1214C*, qui exécute le programme utilisateur, gère les entrées/sorties et assure la communication avec les périphériques. Un module *AI4/AO2* est utilisé pour le traitement des signaux analogiques (4 entrées analogiques et 2 sorties analogiques), tandis qu'un module *DI8/DO8* permet la gestion des signaux numériques (8 entrées digitales et 8 sorties digitales) destinés aux capteurs et actionneurs du procédé.

FIGURE 4.3 – Simulateur — banc de tests / HMI

Description du simulateur : Le simulateur, équipé d'un panneau tactile, est utilisé pour faire fonctionner et tester le système. Il comprend les composants suivants :

- **14 commutateurs** : ces commutateurs servent à simuler des entrées digitales. Le commutateur `I0.1` est configuré comme un contact **NC** (normally closed).
- **10 LED** : utilisées pour visualiser l'état des sorties et des signaux de diagnostic.
- **Potentiomètre rotatif** : permet le réglage manuel ou la simulation de signaux d'entrées analogiques (par exemple poid des pièces).

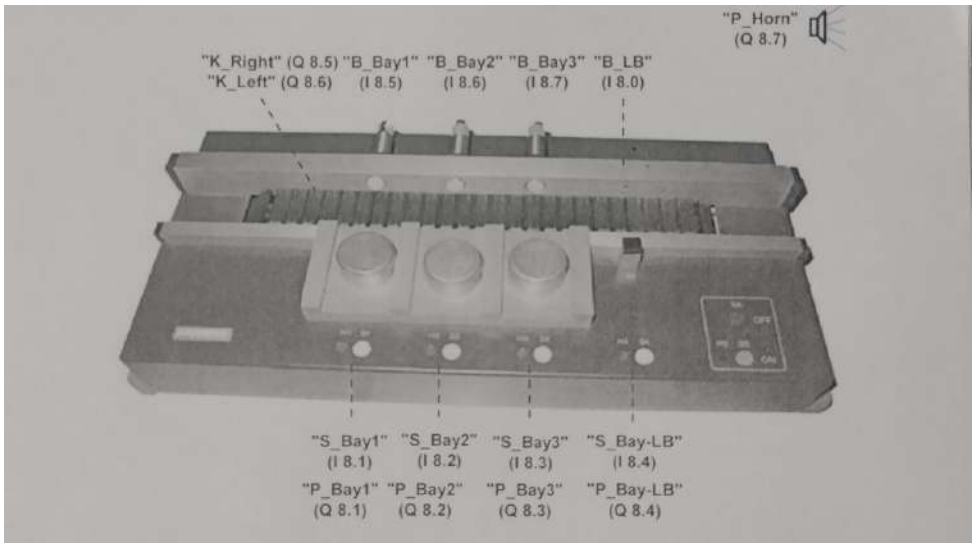
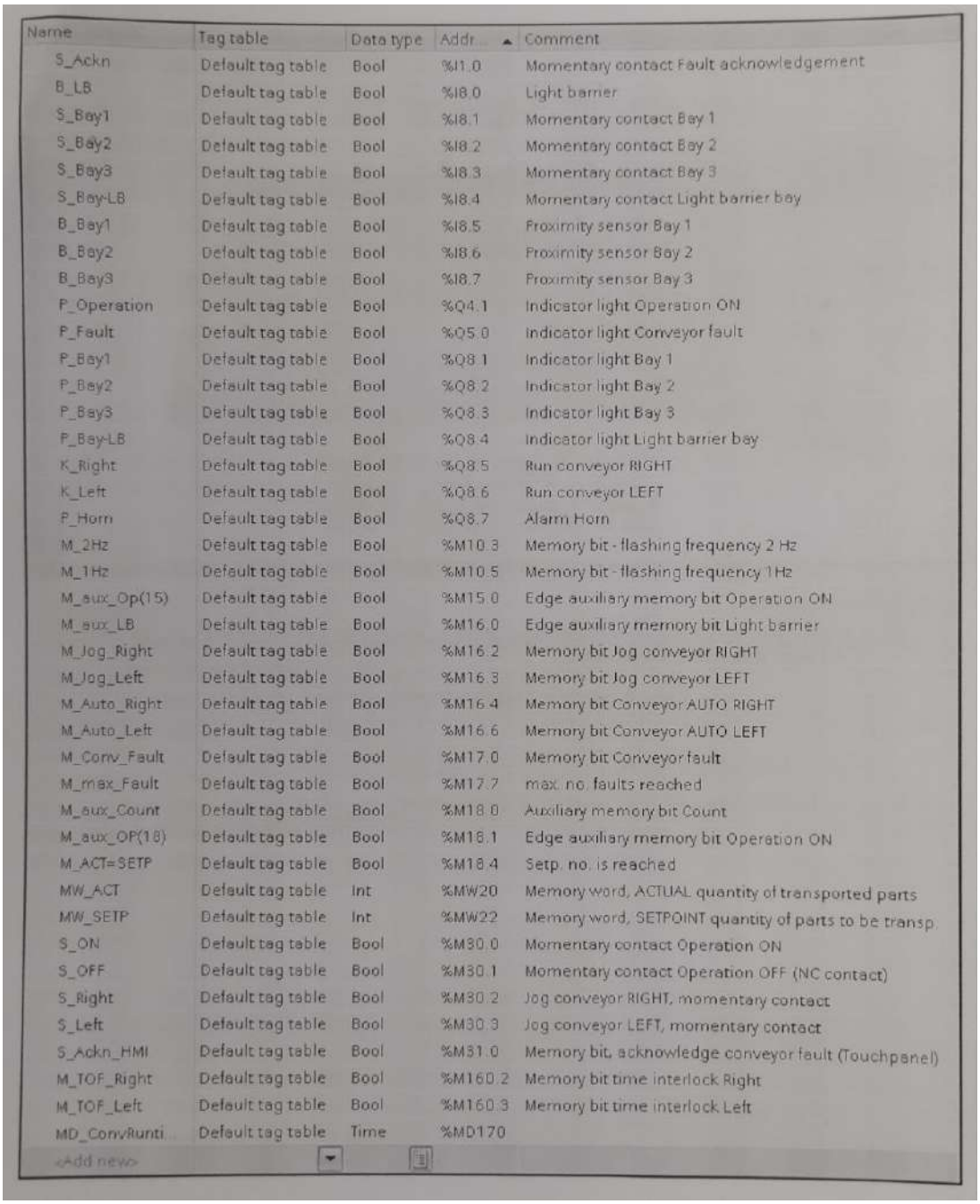


FIGURE 4.4 – Convoyeur — élément mécanique

La photo montre les capteurs et les actionneurs du modèle du convoyeur ainsi que les adresses E / S avec lesquelles ils sont câblés



Name	Tag table	Data type	Addr...	Comment
S_Ackn	Default tag table	Bool	%I1.0	Momentary contact Fault acknowledgement
B_LB	Default tag table	Bool	%I8.0	Light barrier
S_Bay1	Default tag table	Bool	%I8.1	Momentary contact Bay 1
S_Bay2	Default tag table	Bool	%I8.2	Momentary contact Bay 2
S_Bay3	Default tag table	Bool	%I8.3	Momentary contact Bay 3
S_Bay-LB	Default tag table	Bool	%I8.4	Momentary contact Light barrier bay
B_Bay1	Default tag table	Bool	%I8.5	Proximity sensor Bay 1
B_Bay2	Default tag table	Bool	%I8.6	Proximity sensor Bay 2
B_Bay3	Default tag table	Bool	%I8.7	Proximity sensor Bay 3
P_Operation	Default tag table	Bool	%Q4.1	Indicator light Operation ON
P_Fault	Default tag table	Bool	%Q5.0	Indicator light Conveyor fault
P_Bay1	Default tag table	Bool	%Q8.1	Indicator light Bay 1
P_Bay2	Default tag table	Bool	%Q8.2	Indicator light Bay 2
P_Bay3	Default tag table	Bool	%Q8.3	Indicator light Bay 3
P_Bay-LB	Default tag table	Bool	%Q8.4	Indicator light Light barrier bay
K_Right	Default tag table	Bool	%Q8.5	Run conveyor RIGHT
K_Left	Default tag table	Bool	%Q8.6	Run conveyor LEFT
P_Horn	Default tag table	Bool	%Q8.7	Alarm Horn
M_2Hz	Default tag table	Bool	%M10.3	Memory bit - flashing frequency 2 Hz
M_1Hz	Default tag table	Bool	%M10.5	Memory bit - flashing frequency 1Hz
M_aux_Op(15)	Default tag table	Bool	%M15.0	Edge auxiliary memory bit Operation ON
M_aux_LB	Default tag table	Bool	%M16.0	Edge auxiliary memory bit Light barrier
M_Jog_Right	Default tag table	Bool	%M16.2	Memory bit Jog conveyor RIGHT
M_Jog_Left	Default tag table	Bool	%M16.3	Memory bit Jog conveyor LEFT
M_Auto_Right	Default tag table	Bool	%M16.4	Memory bit Conveyor AUTO RIGHT
M_Auto_Left	Default tag table	Bool	%M16.6	Memory bit Conveyor AUTO LEFT
M_Conv_Fault	Default tag table	Bool	%M17.0	Memory bit Conveyor fault
M_max_Fault	Default tag table	Bool	%M17.7	max. no. faults reached
M_aux_Count	Default tag table	Bool	%M18.0	Auxiliary memory bit Count
M_aux_OP(18)	Default tag table	Bool	%M18.1	Edge auxiliary memory bit Operation ON
M_ACT=SETP	Default tag table	Bool	%M18.4	Setp. no. is reached
MW_ACT	Default tag table	Int	%MW20	Memory word, ACTUAL quantity of transported parts
MW_SETP	Default tag table	Int	%MW22	Memory word, SETPOINT quantity of parts to be transp.
S_ON	Default tag table	Bool	%M30.0	Momentary contact Operation ON
S_OFF	Default tag table	Bool	%M30.1	Momentary contact Operation OFF (NC contact)
S_Right	Default tag table	Bool	%M30.2	Jog conveyor RIGHT, momentary contact
S_Left	Default tag table	Bool	%M30.3	Jog conveyor LEFT, momentary contact
S_Ackn_HMI	Default tag table	Bool	%M31.0	Memory bit, acknowledge conveyor fault (Touchpanel)
M_TOF_Right	Default tag table	Bool	%M160.2	Memory bit time interlock Right
M_TOF_Left	Default tag table	Bool	%M160.3	Memory bit time interlock Left
MD_ConvRunti...	Default tag table	Time	%MD170	
<Add new>				

FIGURE 4.5 – Table des PLC Tags dans TIA Portal

Description de la table des PLC Tags : La table des *PLC Tags* regroupe l'ensemble des variables utilisées dans le programme automate. Elle permet de définir les noms symboliques, les adresses (entrées, sorties, mémoires), les types de données ainsi que les commentaires associés. Cette organisation facilite la programmation, la lecture du programme et le diagnostic du système dans le TIA Portal.

4.2 Mise en service du matériel et des logiciels

4.2.1 Connexion en ligne et adressage réseau

Connexion en ligne : Si une connexion en ligne doit être établie entre l'appareil de programmation (PG) et la CPU, *les deux périphériques doivent recevoir le même masque de sous-réseau* et des adresses IP situées dans *le même sous-réseau*. Sans cela, la communication directe ne sera pas possible sans routage intermédiaire.

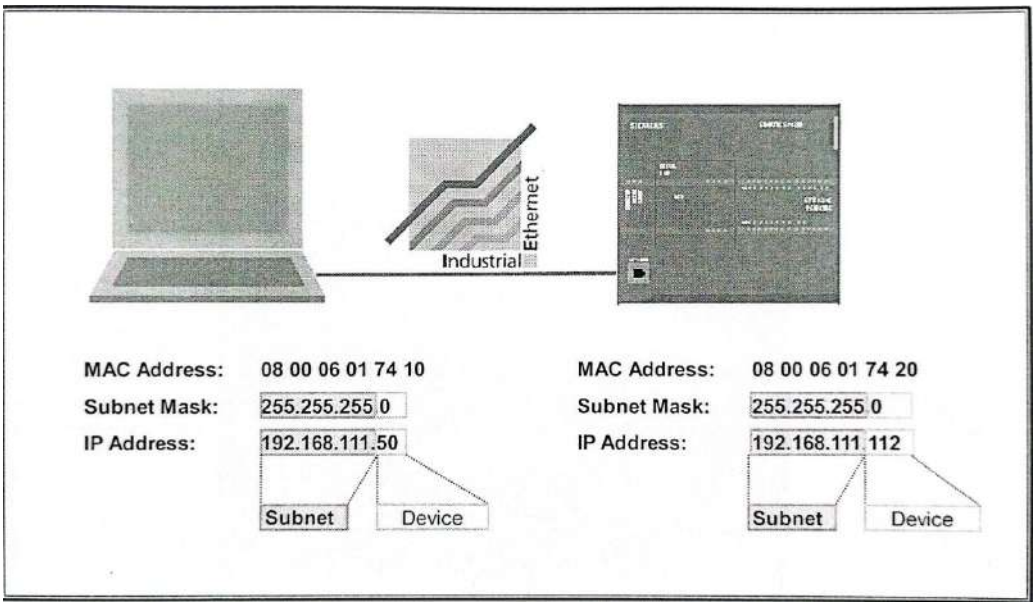


FIGURE 4.6 – Connexion en ligne via Industrial Ethernet (liaison entre PG et CPU)

Protocole Internet (IP)

Le *Internet Protocol (IP)* est à la base des réseaux TCP/IP. Il crée des datagrammes (ou paquets) adaptés au protocole Internet et gère leur transport dans le sous-réseau local ou leur routage vers d'autres sous-réseaux.

Adresses IP

Les adresses IP sont attribuées aux *interfaces réseau* (chaque interface reçoit sa propre adresse). Une adresse IP comprend 4 octets, notés en décimal et séparés par des points (ex : `192.168.1.10`).

Élément	Exemple
Adresse IP de l'appareil	`192.168.1.10`
Masque de sous-réseau	`255.255.255.0`
Adresse réseau (extrait)	`192.168.1.0`
Adresse broadcast (ex.)	`192.168.1.255`

TABLE 4.1 – Exemple d'adressage IP et masque.

Adresse MAC

Chaque interface Ethernet possède une adresse matérielle (MAC) unique attribuée par le fabricant. La MAC identifie l'interface dans le réseau local et est stockée sur la carte réseau.

Masque de sous-réseau

Le masque de sous-réseau permet de diviser l'adresse IP en *adresse réseau* et *adresse d'appareil*. Deux périphériques communiquent directement si leurs adresses IP appartiennent au même réseau défini par le masque.

Affectation d'une adresse IP pour la PG (poste de programmation)

L'adresse IP de la PG (poste de programmation) doit être configurée dans le même sous-réseau que la CPU pour permettre la connexion en ligne. Vérifie également qu'aucune autre machine n'utilise la même adresse (conflit d'IP).

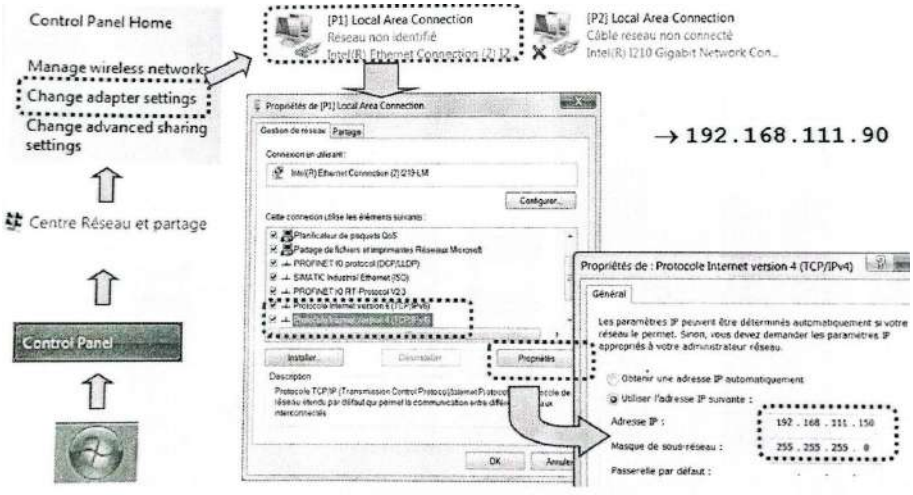


FIGURE 4.7 – Réglage de l'adresse IP sur la PG (exemple d'interface de configuration).

Remarque pratique : exemple d'adressage cohérent :

— CPU : `192.168.1.20` / Masque : `255.255.255.0`

— PG : `192.168.1.10` / Masque : `255.255.255.0`

Les deux adresses appartiennent au même réseau `192.168.1.0/24` et la connexion en ligne pourra s'établir directement.

4.2.2 Mise en service de la valise S7-1214 avec pupitre

Objectif

La valise de formation S7-1214 équipée d'un pupitre tactile a été mise en service. Le projet de base fourni par le formateur a été récupéré sur une clé USB, adapté au matériel présent, compilé et chargé dans la CPU et dans le pupitre.

Étapes réalisées

1. Récupération du projet fourni sur clé USB et ouverture dans TIA Portal.
2. Enregistrement local du projet.
3. Compilation du projet (vérification erreurs/warnings).
4. Chargement de la configuration matérielle et du programme utilisateur dans la **CPU** du PLC1.
5. Réglage de l'adresse IP sur le pupitre (touchpanel) et transfert du projet vers le pupitre.
6. Tests fonctionnels du pupitre et du programme sur la CPU — résultats : **OK**.

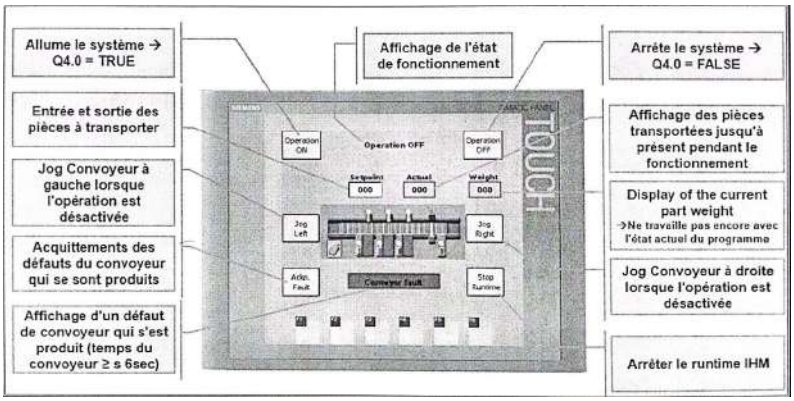


FIGURE 4.8 – Affichage du touchpanel — écran principal de commande et supervision

Fonctions opérées depuis le touchpanel

Les opérations et contrôles disponibles via l'interface tactile sont les suivants :

— Boutons **Operation ON** et **Operation OFF** : activation / désactivation du système.

— Boutons **Jog Right** et **Jog Left** : déplacement manuel (pas à pas) du convoyeur dans la direction souhaitée.

— Champ d'entrée **Setpoint (Consigne :quantité)** : doit contenir une valeur > 0 pour lancer une séquence (nombre de pièces à transporter).

— Champ de sortie **Actual** : affiche le nombre de pièces comptabilisées (pièces ayant traversé la barrière photoélectrique).

— Le champ **Weight** ne retourne pas encore de valeurs : la lecture et le traitement des signaux analogiques seront détaillés dans la section suivante 4.3, intitulée *Traitement des valeurs analogiques*.

Logique de fonctionnement et indicateurs

Signalisation des postes : Lorsque le convoyeur est prêt à recevoir une nouvelle pièce, les voyants des postes 1 et 2 s'allument en continu. Si une nouvelle pièce est détectée au poste, le voyant correspondant clignote à **1 Hz** pour indiquer la présence de la pièce et inviter l'opérateur à démarrer la séquence.

Déroulement du transport : Après démarrage via le bouton poussoir du poste, la pièce est transportée jusqu'à ce qu'elle franchisse la barrière lumineuse. Pendant le transport, les voyants des postes clignotent à **2 Hz**.

Arrêt automatique / défaut : Si une séquence de transport dépasse **6 secondes**, le convoyeur est automatiquement arrêté. Le défaut est signalé par :

— un clignotement visible sur l'écran tactile, et

— la LED `P_Fault` du simulateur (sortie `Q5.0`).

La séquence ne peut être relancée qu'après acquittement du défaut via le bouton tactile approprié ou via le bouton poussoir simulateur `S_Ackn` (entrée `I1.0`).

Comptage des pièces : Les pièces sont comptées lorsqu'elles traversent la barrière photoélectrique ; le compteur est affiché dans **Actual**. Lorsque la valeur saisie dans **Setpoint** est atteinte, l'état 'consigne atteinte' est signalé par la lampe `P_Bay-LB` (sortie `Q8.4`). Une nouvelle séquence n'est possible qu'après acquittement via le bouton poussoir `S_Bay-LB` (entrée `I8.4`).

Tableau récapitulatif des E/S pertinentes

Adresse	Symbole	Fonction
I1.0	S_Ackn	Bouton d'acquittement (simulateur)
Q5.0	P_Fault	LED défaut (clignote en cas d'erreur >6s)
I8.4	S_Bay-LB	Bouton poussoir d'acquittement (entrée)
Q8.4	P_Bay-LB	Voyant indiquant que la consigne est atteinte
(barrière photo)	—	Détecteur qui incrémente le compteur lors du franchissement (entrée digitale)

TABLE 4.2 – Adresses E/S utilisées dans les tests fonctionnels

4.3 Traitement des valeurs analogiques

4.3.1 Description de la tâche

Dans ce chapitre, la conversion et le traitement des signaux analogiques sont traités. Pour cela, une tension est définie et lue sur le potentiomètre du simulateur. Cette tension simule les valeurs de poids de la pièce.

Il vous incombera de convertir les valeurs lues toutes les **250 ms** dans l'interruption cyclique en valeurs de poids comprises entre **0 kg et 500 kg** en utilisant les opérations `NORM_X` et `SCALE_X`.

Le poids n'est valable que dans la plage :

$$100 \text{ kg} \leq \text{Poids} \leq 400 \text{ kg}$$

Si le poids de la pièce dépasse ou tombe en dessous de ces limites, la pièce est considérée comme invalide et aucune autre séquence de transport ne peut être démarrée (les LED Bay restent éteintes et le mouvement du convoyeur vers la droite ne peut pas être démarré).

De plus, ce chapitre permet d'apprendre la procédure à suivre en cas de défaillance d'un canal d'un module analogique afin d'obtenir des informations détaillées sur l'événement de panne.

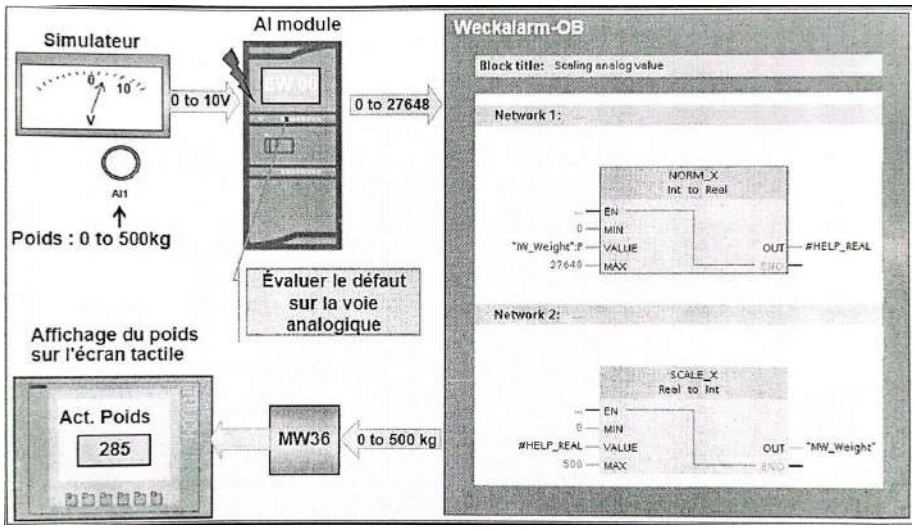


FIGURE 4.9 – Principe du traitement des valeurs analogiques

4.3.2 Conversion de la valeur analogique et affichage sur l'écran tactile

La valeur analogique convertie, représentant le poids de la pièce (0 à 500 kg), est enregistrée dans le mot mémoire :

`MW36` (`MW_Weight`)

La validité du poids est vérifiée à l'aide de la condition suivante :

— Si 100 kg $\leq$ `MW36` $\leq$ 400 kg $\rightarrow$ `M35.0` (`M_Weight_OK`) = `TRUE`

— Sinon $\rightarrow$ `M35.0` (`M_Weight_OK`) = `FALSE`

La mémoire binaire `M35.0` est déjà liée au champ d'E/S « **Act. Weight** » sur le touchpanel et influence la couleur de son arrière-plan.

Si `M35.0` fournit la valeur `FALSE`, les voyants lumineux des postes du convoyeur doivent rester éteints et aucune nouvelle séquence de transport ne peut être démarrée. Les verrouillages nécessaires dans **FC14** et **FC16** doivent être programmés.

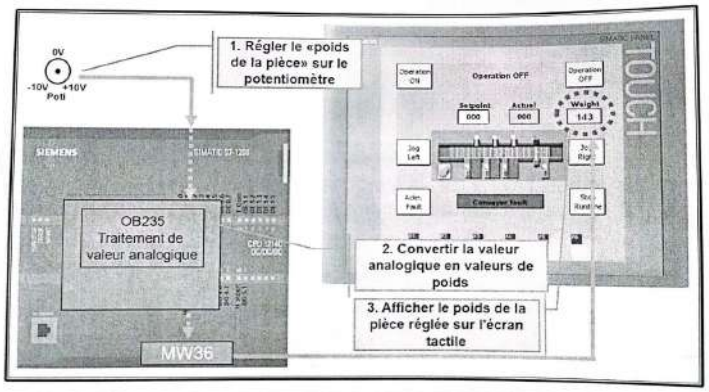


FIGURE 4.10 – Affichage du poids et indication de validité sur le touchpanel

4.3.3 Implémentation et tests

Un traitement de valeur analogique a été programmé dans une interruption cyclique **OB235**. La valeur analogique convertie est traitée, validée et stockée dans `MW36`, tandis que l'état de validité est attribué à `M35.0`.

Les blocs fonctionnels **FC14** et **FC16** ont été verrouillés dans l'OB235 afin d'empêcher toute séquence de transport lorsque le poids est invalide.

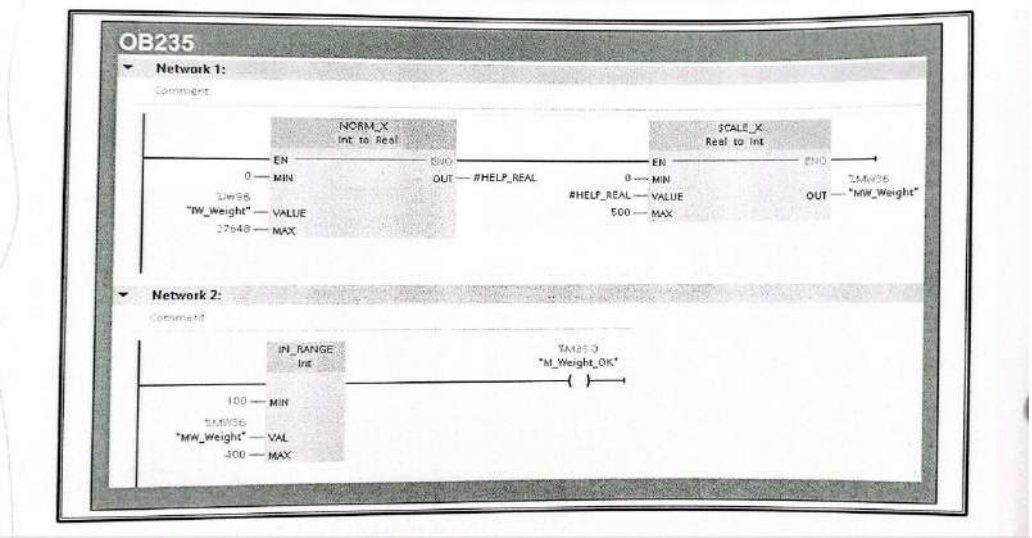


FIGURE 4.11 – Programmation du traitement analogique dans l'OB235

Après le chargement des blocs dans la CPU, les tests ont été effectués sur le touchpanel :

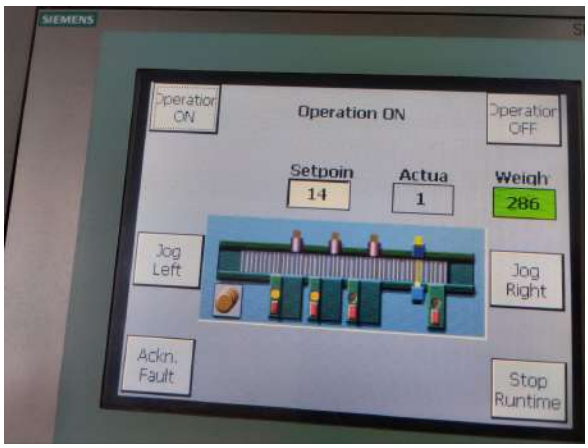
— Les valeurs de poids sont affichées correctement.

— La couleur d'arrière-plan change lorsque le poids est incorrect.

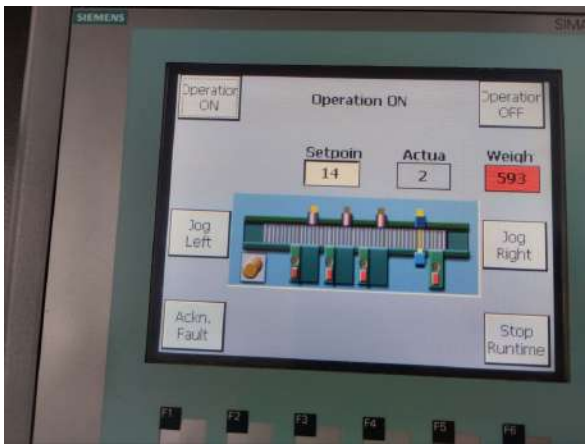
— En cas de poids non valide, les voyants des postes restent éteints.

— Le moteur du convoyeur ne peut plus être démarré.

Ces essais confirment le bon fonctionnement du traitement des valeurs analogiques et des sécurités associées.



(a) Affichage HMI — poids valide : **286 kg** (dans la plage 100–400 kg)



(b) Affichage HMI — poids non valide : **593 kg** ($>$ 400 kg) — indication d'alerte (arrière-plan changé)

FIGURE 4.12 – Exemples d'affichage sur le touchpanel : (a) poids valide affiché, (b) poids hors plage avec signalisation visuelle.

4.4 Blocs de données

4.4.1 DB_Parts (DB99)

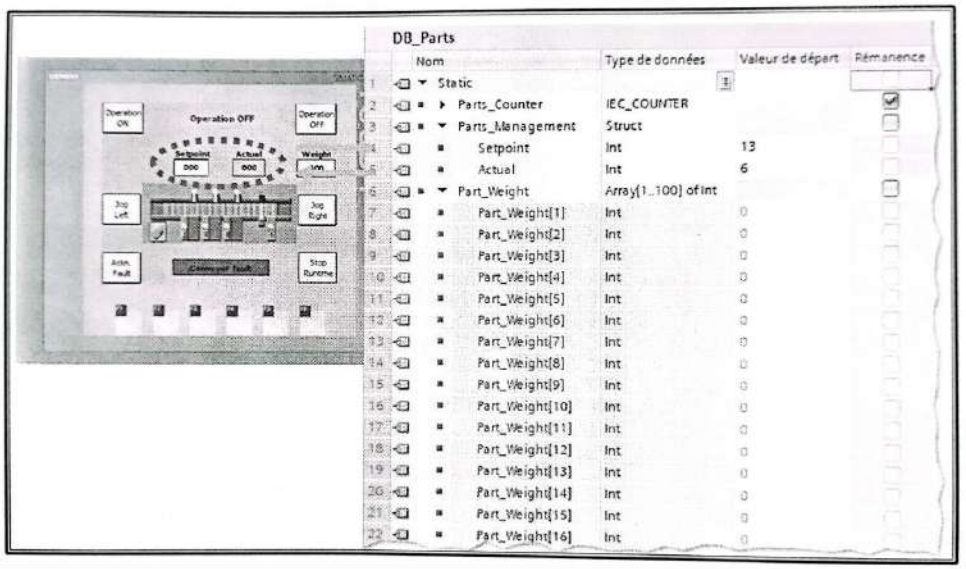


FIGURE 4.13 – Présentation du bloc de données `DB_Parts` (DB99)

Fonction existante Pour gérer le point de consigne et les quantités réelles, les variables mémoires `MW_SETP` (`MW22`) et `MW_ACT` (`MW20`) sont utilisées. Celles-ci sont liées aux champs d'E/S correspondants sur l'écran tactile.

Description de la tâche Pour la gestion avancée des poids de pièces, un bloc de données `DB_Parts` (DB99) doit être créé. Dans ce bloc, on définit :

— une variable de type **Structure** nommée `Parts_Management` contenant au moins les éléments `Setpoint` et `Actual`,

— une variable **Array** nommée `Part_Weight` pour stocker les poids individuels.

Les éléments `Parts_Management.Setpoint` et `Parts_Management.Actual` remplaceront les variables mémoires actuelles (`MW22` et `MW20`) dans la fonction `FC_Count` (FC18). De plus, une variable rémanente du type `IEC_COUNTER` doit être créée dans ce bloc de données afin de rendre le compteur rémanent depuis `FC_Count`.

Réalisation (procédure)

1. Créer et déclarer le bloc de données `DB_Parts` (DB99) dans `PLC_1` → `Program blocks`.
2. Définir la `STRUCT Parts_Management` et l'array `Part_Weight` (par ex. `ARRAY[1..100] OF INT`).

3. Ajouter la variable rémanente de type `IEC_COUNTER` dans DB99.
4. Ouvrir `FC_Count` (double-clic dans `PLC_1` → `Program blocks`).
5. Afficher l'éditeur divisé (horizontale/verticale) pour faciliter le glisser-déposer.
6. Par glisser-déposer, remplacer les usages de `MW_SETP` par `DB_Parts.Parts_Management.Setpoint` (paramètre PV du compteur).
7. De même, remplacer `MW_ACT` par `DB_Parts.Parts_Management.Actual` (paramètre CV du compteur).
8. Rendre le compteur IEC rémanent (paramétrage sur le compteur / déclaration rémanente).
9. Enregistrer et charger le projet modifié dans la CPU.

Résultat Après le transfert vers la CPU et la visualisation du `DB_Parts`, la valeur `Actual` est conservée même après redémarrage du processeur (compteur rémanent). La consigne a pour valeur de départ `5`.

Remarque en attente Actuellement, les champs d'E/S sur l'écran tactile sont encore liés aux anciennes variables mémoires (`MW22`, `MW20`) et n'affichent pas les nouvelles valeurs issues du DB. Ce comportement sera corrigé dans l'exercice suivant : *Mise à jour des tags d'interface IHM et transfert vers le touchpanel.*

4.4.2 Archivage de `Part_Weight` dans DB_Parts en utilisant `FieldWrite`

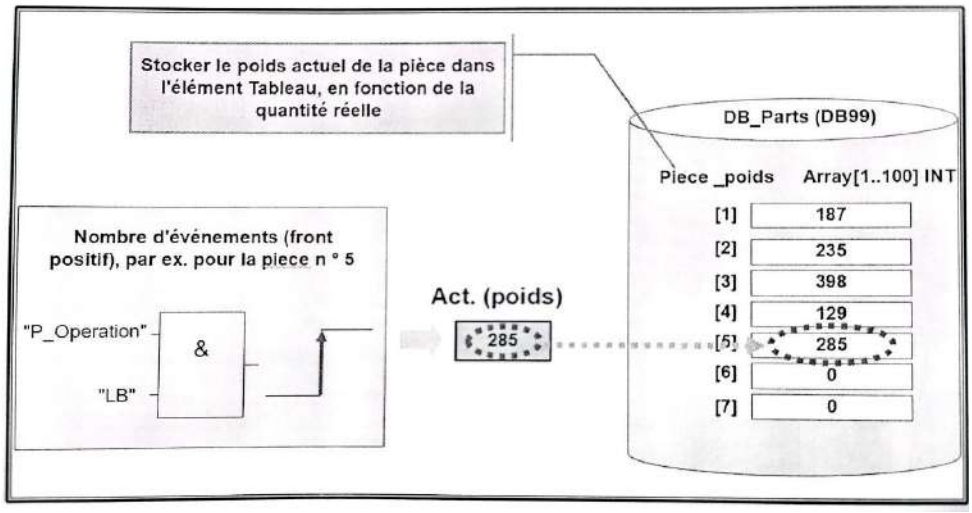


FIGURE 4.14 – Schéma fonctionnel : archivage des poids dans `DB_Parts.Part_Weight` via `FieldWrite`

Fonction existante Pendant l'opération, les pièces placées sur les postes 1 ou 2 sont comptées au compteur IEC `DB_Parts.Parts_Counter`. Le poids relatif de chaque pièce est déterminé par la conversion analogique et validé (voir Section 4.3).

Description de la tâche Les poids individuels des pièces doivent être stockés dans les éléments du tableau `DB_Parts.Part_Weight` à chaque passage de la barrière photoélectrique. L'instruction `FieldWrite` doit être utilisée pour écrire la valeur dans l'index approprié du tableau. L'index à écrire est donné par `DB_Parts.Parts_Management.Actual` (le numéro de pièce courant).

Fonction réutilisable `FC_Ind_Weight` (FC35) Créer une fonction réutilisable `FC_Ind_Weight` (FC35) avec l'interface suivante :

- **IN**
  - `Store_Weight` : BOOL — déclenche l'enregistrement du poids
  - `Act_Weight` : INT — valeur de poids à stocker
  - `Part_No` : INT — index actuel (numéro de pièce)
- **OUT**
  - `Weight_Storage` : ARRAY[1..100] OF INT — paramètre pour DB-Array (liaison à `DB_Parts.Part_Weight`)

Procédure de programmation (extrait)

1. Dans `FC_Ind_Weight`, programmer une instruction `CONV` pour convertir la variable locale `Part_No` (INT) en `DINT` — ceci car le paramètre `INDEX` de `FieldWrite` requiert un `DINT`.
2. Programmer l'appel à `FieldWrite` :
   - Paramètre `DB` : `DB_Parts`
   - Paramètre `FIELD` : adresse du tableau (`Part_Weight`)
   - Paramètre `INDEX` : valeur convertie (DINT) correspondant à `Part_No`
   - Paramètre `VALUE` : `Act_Weight`
3. Gérer la condition de trigger : n'exécuter `FieldWrite` que si `Store_Weight` = TRUE.
4. Enregistrer la fonction.

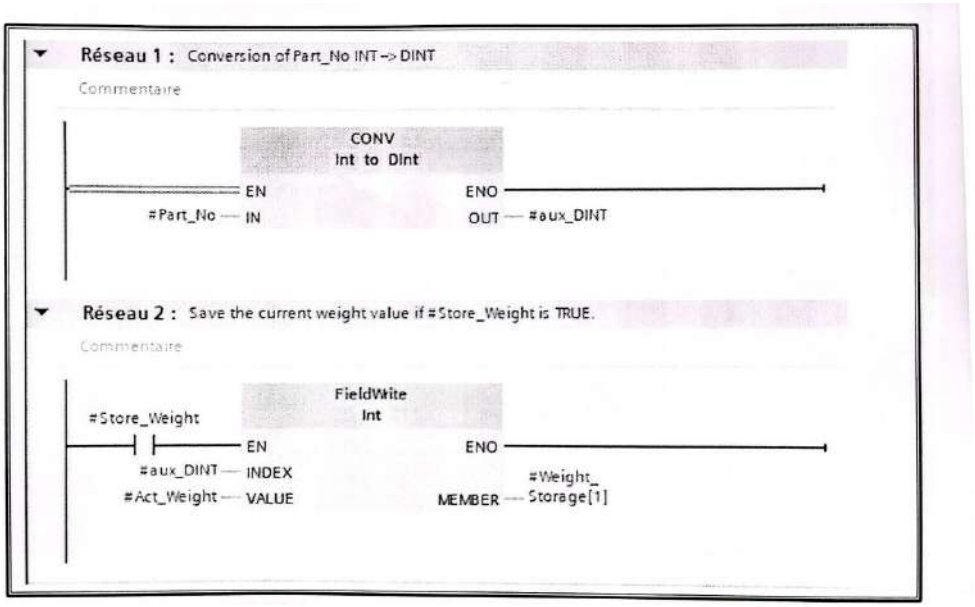


FIGURE 4.15 – Conversion INT→DINT et instruction `FieldWrite`

Intégration et tests

1. Appeler `FC_Ind_Weight` depuis `FC_Count` au moment où une pièce est validée (passage barrière).
2. Transférer la configuration et les blocs vers la CPU.
3. Visualiser `DB_Parts` : vérifier que les valeurs de `Part_Weight` sont écrites séquentiellement.

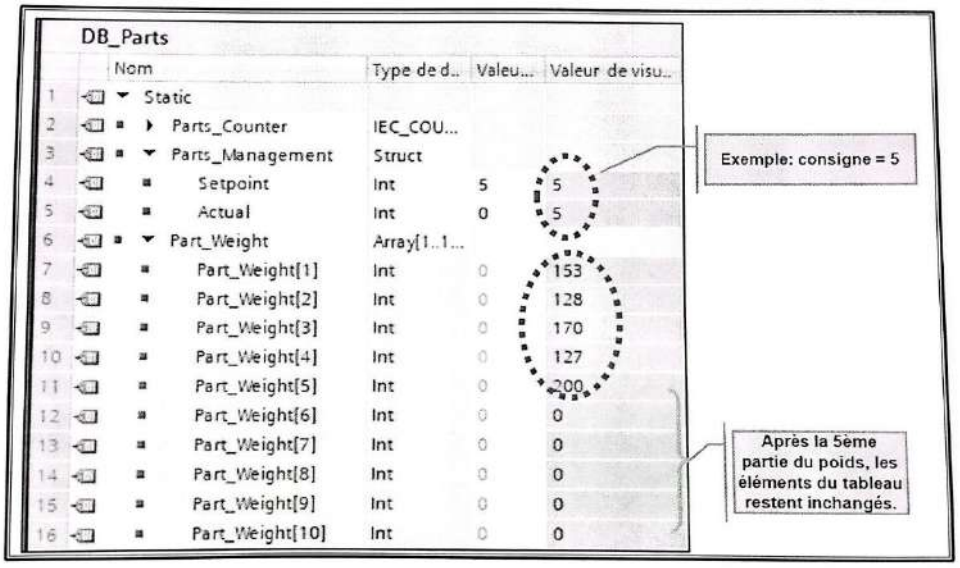


FIGURE 4.16 – Visualisation de `DB_Parts.Part_Weight` après insertion des valeurs

Résultat observé Les valeurs de poids des pièces sont insérées successivement dans les éléments du tableau `Part_Weight` jusqu'à l'atteinte de la consigne. Si une nouvelle

production démarre après réinitialisation de `Actual`, les anciennes valeurs sont écrasées.

4.5 Introduction à la communication industrielle

4.5.1 Description de la tâche : Création d'une liaison "ISO-on-TCP"

Une liaison `ISO-on-TCP` entre les CPU de la zone de formation doit être programmée. Via cette liaison, la variable `ARRAY DB_Parts.Part_Weight` est envoyée du contrôleur `PLC_1` au contrôleur `PLC_2`. Dans `PLC_2`, les données reçues doivent être enregistrées dans le bloc de données `DB_RECEIVE` (variable `ARRAY Receive_buffer`).

L'envoi n'est pas continu afin de minimiser le trafic — il doit se produire dans les mêmes conditions que la sauvegarde des valeurs de poids dans `DB_Parts` (c.-à-d. lors du passage des pièces validées à la barrière photoélectrique).

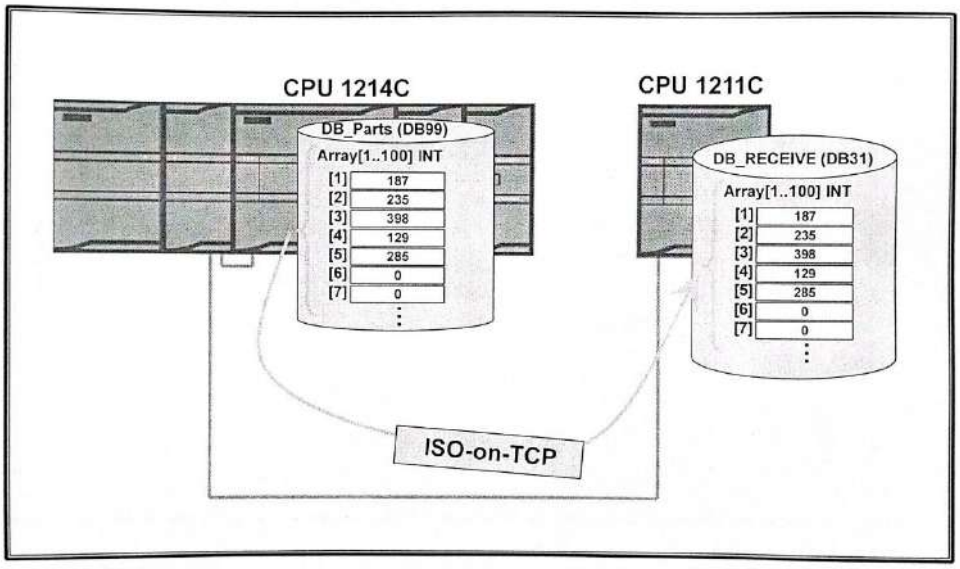


FIGURE 4.17 – Topologie : liaison ISO-on-TCP entre `PLC_1` et `PLC_2`

4.5.2 Programme de communication CPU–CPU et envoi de 200 octets de données

Situation actuelle : les valeurs individuelles des poids des pièces valides sont stockées dans `DB_Parts.Part_Weight` à chaque franchissement de la barrière photoélectrique.

Objectif : envoyer ces valeurs (200 octets au total) de `PLC_1` vers `PLC_2` uniquement lors des mêmes événements qui déclenchent l'archivage (pour réduire le trafic).

4.5.3 Blocs utilisés

Contrôleur	Bloc / rôle
PLC_1	`FC_Send` (FC30) — appel dans `FC_Count` (FC18) ; utilise `TSEND_C`
PLC_2	`FC_Receive` (FC31) — appelé dans `MAIN` (OB1) ; utilise `TRCV_C`
PLC_2	`DB_RECEIVE` (DB31) — contient `Receive_buffer : ARRAY[1..100] OF INT`

TABLE 4.3 – Blocs et affectations pour la communication CPU–CPU

4.5.4 Configuration réseau et matériel

La CPU `PLC_2` (type S7-1211C) a été ajoutée au projet et connectée dans la vue réseau du TIA Portal.
L'adresse IP attribuée à `PLC_2` est : `192.168.111.114`.

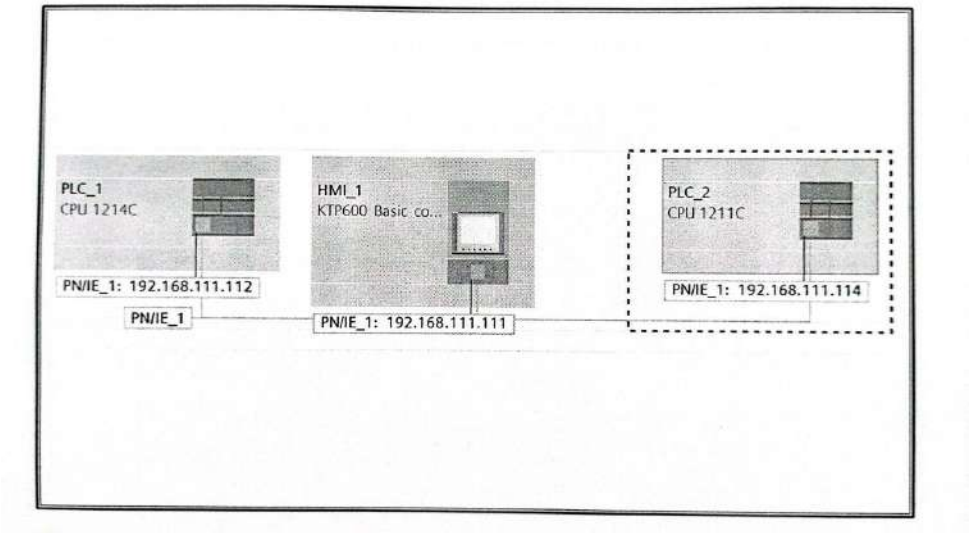


FIGURE 4.18 – Vue réseau du projet — ajout de `PLC_2` et connexion

4.5.5 Création du DB_RECEIVE sur PLC_2

Un nouveau bloc de données `DB_RECEIVE` (DB31) a été créé sur `PLC_2`. Il contient la variable :

```
Receive_buffer : ARRAY[1..100]  OF  INT
```

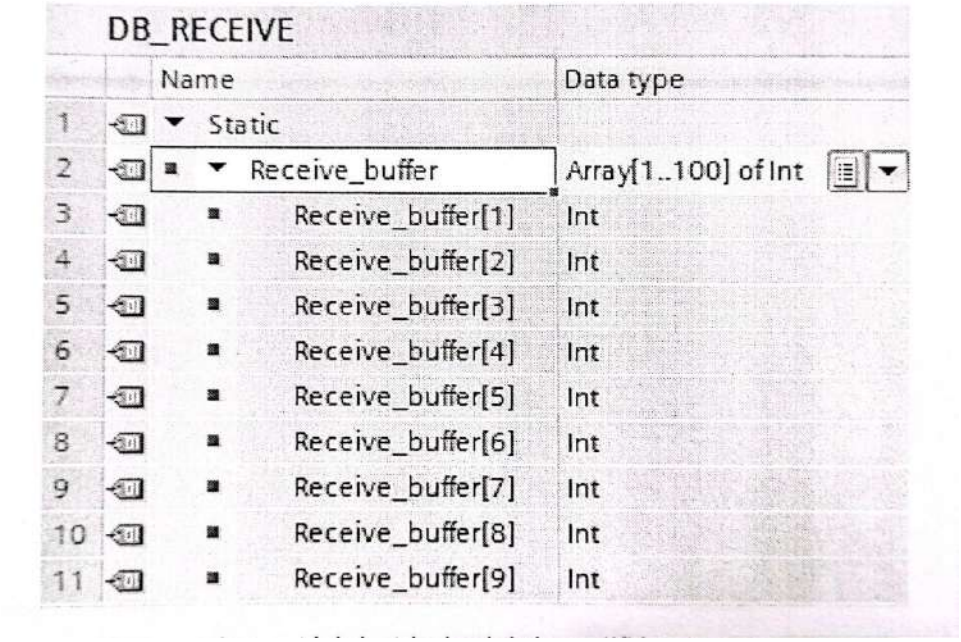


FIGURE 4.19 – Création du bloc `DB_RECEIVE` et déclaration de `Receive_buffer`

Après ces modifications matérielles et logicielles, le projet modifié a été téléchargé dans les CPU concernées.

4.5.6 Programmation de l'envoi (`FC_Send` — PLC_1)

1. Création de `FC_Send` (FC30) dans `PLC_1`.
2. Programmation d'un appel à la fonction système `TSEND_C` pour envoyer un bloc de données (200 octets) vers l'adresse IP de `PLC_2` sur le port configuré.
3. L'appel à `FC_Send` est inséré dans `FC_Count` (FC18) et est déclenché aux mêmes conditions que l'archivage (ex. : `Store_Weight = TRUE` / passage barrière).
4. Préparation du buffer d'envoi (lecture de `DB_Parts.Part_Weight` et empaquetage en octets/structure appropriée avant `TSEND_C`).

4.5.7 Programmation de la réception (`FC_Receive` — PLC_2)

1. Création de `FC_Receive` (FC31) dans `PLC_2`.
2. Programmation d'un appel à la fonction système `TRCV_C` (ou routine équivalente) pour attendre et lire les données entrantes.
3. Écriture des données reçues dans `DB_RECEIVE.Receive_buffer`.
4. Appel de `FC_Receive` depuis `OB1` (MAIN) pour assurer le traitement continu des messages reçus.

4.5.8 Vérification et résultats

— Après chargement sur les deux CPU, la communication est testée en visualisant `DB_RECEIVE` sur `PLC_2`.

— Initialement, aucune donnée n'est affichée. Lorsqu'une pièce valide franchit la barrière photoélectrique, la séquence d'envoi est déclenchée et les **200 octets** sont transmis à `PLC_2`.

— Après réception complète, `Receive_buffer` dans `DB_RECEIVE` affiche les valeurs transférées.

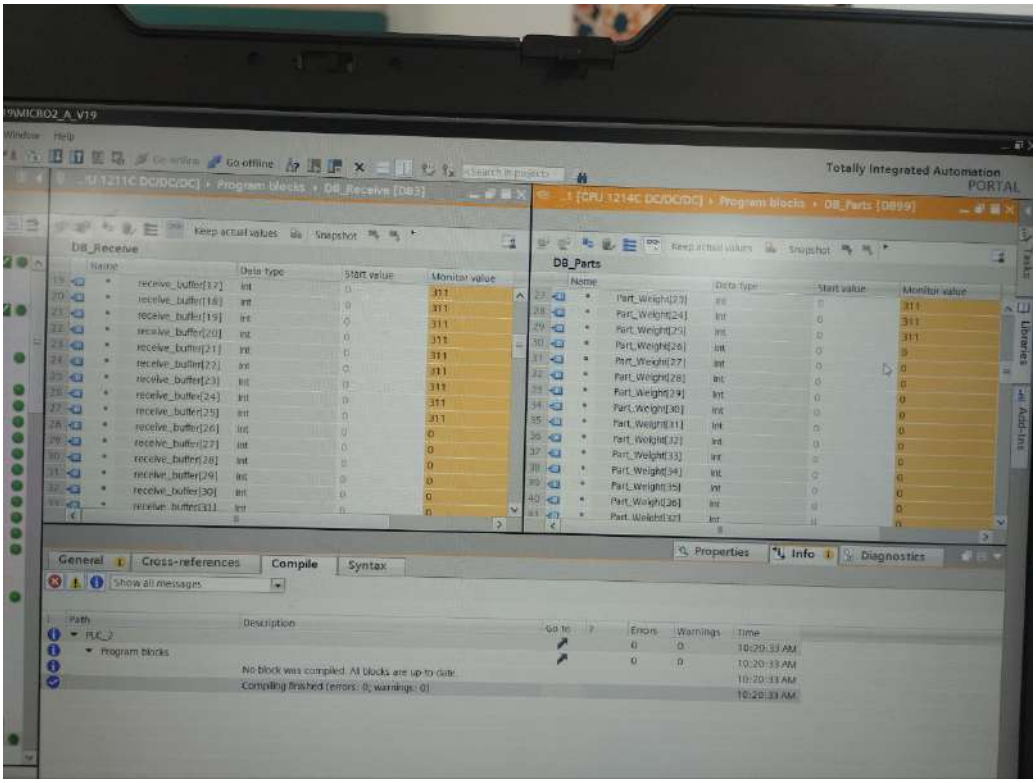


FIGURE 4.20 – Visualisation de `DB_RECEIVE.Receive_buffer` sur `PLC_2` après réception

4.6 Introduction à PROFINET

4.6.1 Descriptif de la tâche : Remplacement d'un module d'E/S TOR par des E/S déportées

Le convoyeur est actuellement contrôlé par le module central `8DI/8DO` du châssis principal. L'objectif est de mettre en œuvre un périphérique **PROFINET** (ET200S) afin que le modèle de convoyeur soit contrôlé par l'ET200S avec les mêmes fonctionnalités, **sans modifier le programme S7** existant.

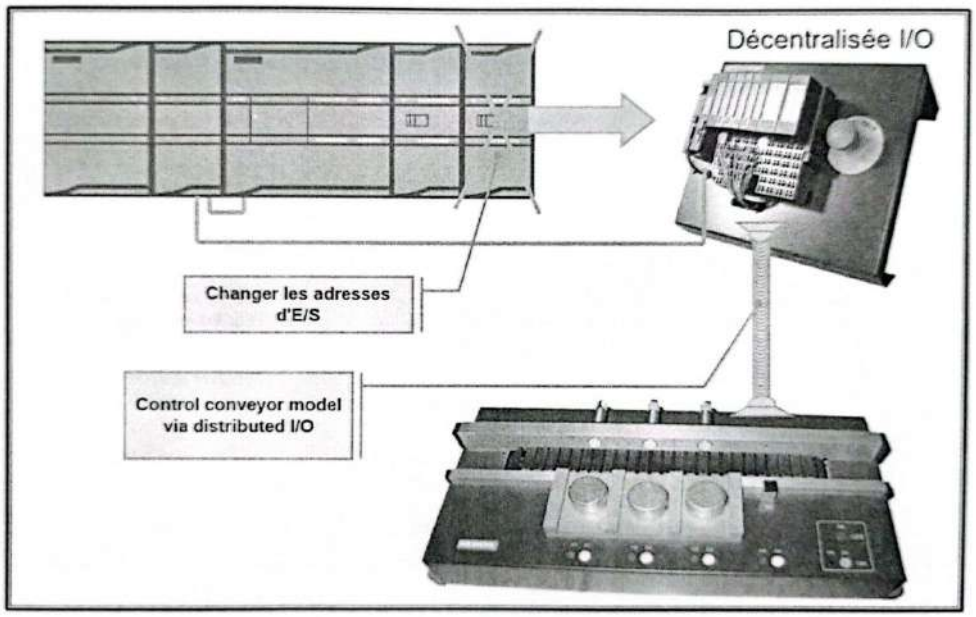


FIGURE 4.21 – Insertion de l'ET200S dans la topologie PROFINET

4.6.2 Situation initiale

Les signaux d'entrée binaires (capteurs, barrières, boutons) du modèle de convoyeur sont mappés sur `IB8` ; les signaux de sortie (voyants, bobines) sont mappés sur `QB8`. Ces adresses sont traitées actuellement par le module central `8DI/8DO`.

4.6.3 Tâche : Contrôle du modèle de convoyeur via l'ET200S

Le but est d'utiliser l'ET200S comme esclave PROFINET pour remplacer le module central 8DI/8DO en réutilisant les mêmes adresses d'E/S (`IB8` / `QB8`) afin d'éviter toute modification du programme utilisateur S7.

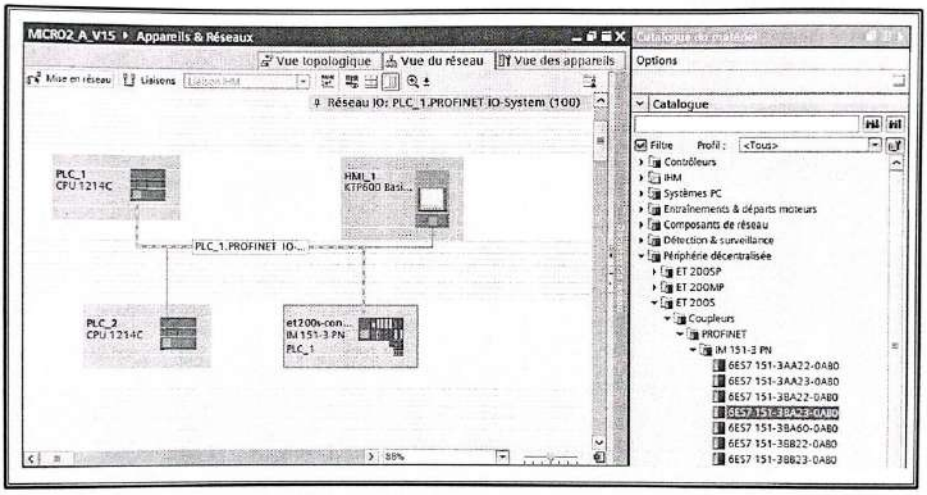


FIGURE 4.22 – ET200S ajouté et connecté au réseau PLC_1 (vue projet)

4.6.4 Étapes réalisées

1. Ajout de l'appareil `ET200S` dans le projet TIA Portal et connexion réseau avec `PLC_1`.
2. Configuration des modules ET200S (choix des modules d'E/S compatibles).
3. Définition des adresses PROFINET et affectation des adresses d'E/S afin que les adresses physiques correspondent aux adresses logiques `IB8` / `QB8`.
4. Modification des adresses du module central `8DI/8DO` : définir la plage d'adresses d'entrée/sortie sur `adresse de début = 88` (voir tableau ci-dessous).
5. Compilation de la configuration matérielle modifiée.
6. Transfert de la configuration matérielle vers la CPU (`PLC_1`) et vers l'ET200S.
7. Débranchement du connecteur du câble du convoyeur du châssis central (PLC1) et branchement sur le connecteur de l'ET200S.
8. Tests fonctionnels : vérification que le programme se comporte comme avant (ON/OFF, Jog, voyants, barrières, etc.).

4.6.5 Adresses d'E/S

Périphérique	Type	Adresse de début
Module central 8DI/8DO	Entrées (IB)	88
Module central 8DI/8DO	Sorties (QB)	88
ET200S (modules correspondants)	Entrées / Sorties	mappées pour correspondre à IB8 / QB8

TABLE 4.4 – Attribution des adresses d'E/S

4.6.6 Procédure détaillée (points clés)

1. Dans la vue de l'appareil `PLC_1`, ouvrir les propriétés du module `8DI/8DO` et noter les adresses actuelles (IB8 / QB8).
2. Insérer l'ET200S dans la vue réseau et ajouter les modules d'E/S correspondants (8DI, 8DO ou modules équivalents).
3. Définir les adresses d'E/S de l'ET200S pour qu'elles correspondent aux adresses IB8 / QB8.
4. Compiler la configuration de périphérique modifiée et résoudre les éventuels avertissements.
5. Télécharger la configuration matérielle sur la CPU (`PLC_1`) et sur l'ET200S.
6. Physiquement transférer le câble du convoyeur (connecteur) vers l'ET200S.
7. Réaliser les tests fonctionnels en mode RUN et vérifier le comportement identique au fonctionnement précédent.

4.6.7 Résultats des tests

Après transfert de la configuration et connexion physique :

— Le programme S7 existant n'a pas nécessité de modification.

— Les fonctions du convoyeur (activation, jog, voyants, comptage) ont été vérifiées et fonctionnent correctement via l'ET200S.

4.6.8 Remarque pratique — E/S déportées à proximité du procédé

Il est souvent préférable d'installer la station d'E/S déportées (par exemple `ET200S`) *au plus près* du système ou de la machine que l'on souhaite contrôler lorsque celui-ci se trouve à distance du châssis central. Cette approche présente plusieurs avantages opérationnels et économiques :

— **Réduction du câblage faible tension** : les liaisons capteurs/actionneurs (analogiques ou numériques) restent courtes, ce qui diminue le coût et la complexité du câblage sur le site.

— **Meilleure immunité au bruit** : les signaux analogiques et numériques parcourent moins de longueur, réduisant les perturbations électromagnétiques et les erreurs de lecture.

— **Maintenance simplifiée** : les points d'interconnexion étant localisés près du procédé, les interventions sont plus rapides et localisables.

— **Scalabilité et modularité** : on peut ajouter plusieurs stations ET200 successives pour couvrir une grande zone sans augmenter la complexité du châssis central.

— **Optimisation des coûts** : diminution des coûts d'installation (moins de câbles, moins de cheminements) et possibilité d'utiliser des câbles Ethernet industriels pour la liaison longue distance.

Points d'attention lors d'une architecture déportée :

— prévoir une **alimentation locale** et une mise à la terre adaptée pour la station ET200 ;

— dimensionner la **liaison réseau** (switch industriel, fibre optique si nécessaire) en tenant compte de la latence et des exigences temps réel de `PROFINET` ;

— conserver le même mapping d'adresses (IB/QB) ou mettre à jour soigneusement les tags dans le projet pour éviter toute modification du programme S7.

En pratique, liaisons Ethernet industrielles (câble blindé, switches industriels, ou fibre pour très longues distances) sont utilisées pour relier la `CPU` au(x) `ET200` déporté(s), ce qui permet de commander efficacement des systèmes situés loin du châssis sans sacrifier la performance ni la fiabilité.

4.7 Tags et messages dans HMI

4.7.1 Descriptif de la tâche : Valeur de sortie analogique, configuration des messages et synchronisation de l'heure avec la CPU

on a configuré plusieurs éléments sur l'IHM :

1. Un champ de sortie affichant le temps de surveillance du transport (variable API `MD_TranspTime`).
2. Une alarme discrète « Consigne quant. Atteinte ! » liée au bit `M33.2` dans le mot `MW_Messages` (`MW32`).
3. Une alarme analogique affichant la valeur du poids (`MW_Weight`) lorsque celle-ci sort de la plage valide.
4. La synchronisation cyclique de l'heure CPU vers le touchpanel (fonction `RD_LOC_T` et pointeur de zone IHM).

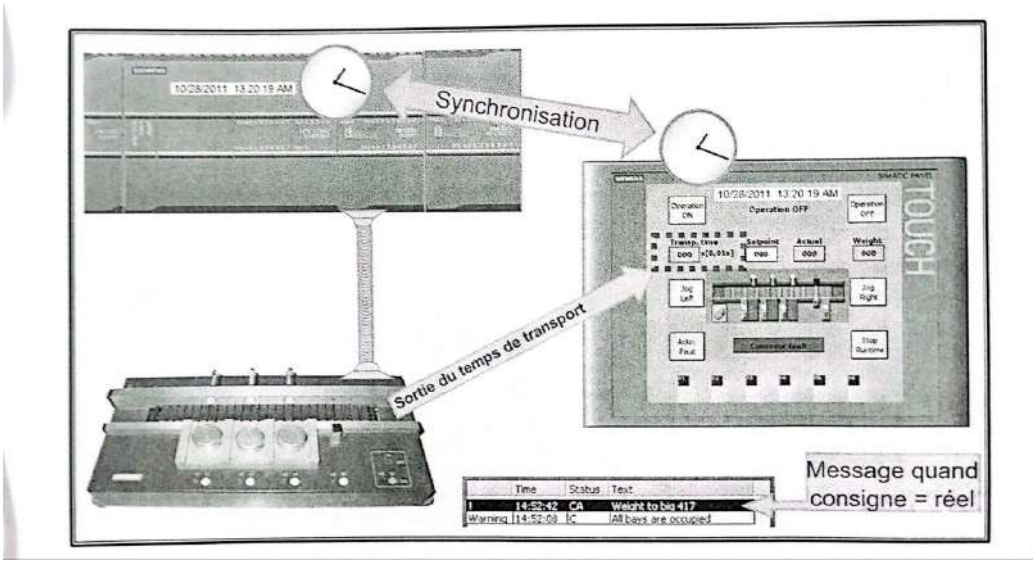


FIGURE 4.23 – Vue d'ensemble du projet IHM — écrans et fenêtres de message

4.7.2 Saisie / affichage du temps de transport (champ de sortie)

Sur l'écran `Convoyeur`, nous avons créé un nouveau *champ de sortie* et nous l'avons lié à la variable API `MD_TranspTime` afin d'afficher le temps de surveillance du transport écoulé. Nous avons ajusté les propriétés d'affichage (format, unités) et ajouté les libellés nécessaires.

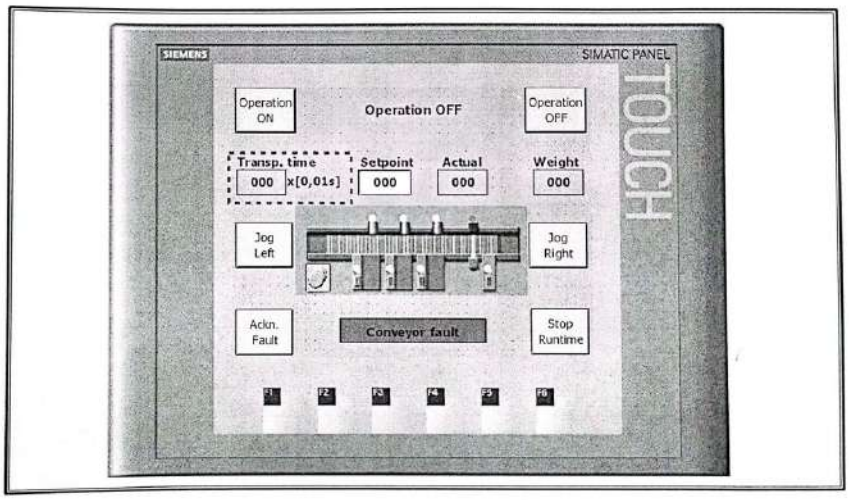


FIGURE 4.24 – Configuration du champ d'E/S affichant `MD_TranspTime` sur l'écran *Convoyeur*

4.7.3 Configuration d'une alarme TOR (discrète) et d'alarmes analogiques

Alarme discrète — Consigne atteinte Une alarme discrète intitulée *Consigne quant. Atteinte !* a été créée et son bit de message a été associé au bit mémoire `M33.2` (situé dans `MW_Messages / MW32`). Lorsqu'une condition `MW_SETP = MW_ACT` a été détectée, le bit `M33.2` a été positionné à `TRUE` afin de déclencher l'alarme.

Alarmes analogiques — poids hors plage Des alarmes analogiques ont été configurées pour les cas *poids < 100 kg* et *poids > 400 kg*. Lors du déclenchement, le texte d'alarme ainsi que la valeur actuelle ont été affichés (liaison de l'affichage à `MW_Weight`). L'IHM disposait déjà d'une fenêtre d'affichage des alarmes pour les classes *Erreurs*, *Avertissements* et *Système*.

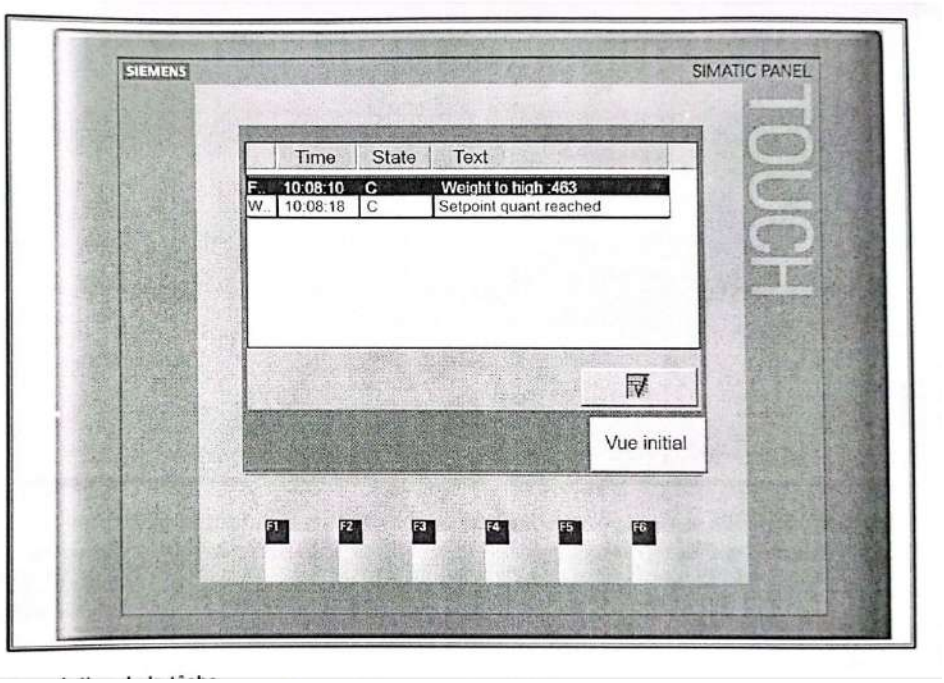


FIGURE 4.25 – Configuration des alarmes discrètes et analogiques dans l'éditeur IHM

Résultat

— À la consigne atteinte : message discret « Consigne quant. Atteinte ! » apparaît dans la fenêtre d'alarmes.

— Si le poids sort de la plage valide : message analogique avec la valeur `MW_Weight` s'affiche (texte et valeur).

4.7.4 Synchronisation de l'horloge CPU → Touchpanel

But Transférer périodiquement l'heure locale de la CPU vers l'IHM afin que le touch-panel affiche l'heure exacte de l'automate.

Partie PLC

1. Le bloc de données `DB_HMI_Sync` (DB11) a été créé.
2. Dans DB11, la balise `Date_Time` de type `DTL` a été créée.
3. La fonction `FC_HMI_Sync` (FC11) a été créée.
4. Dans `FC_HMI_Sync`, la fonction `RD_LOC_T` a été appelée afin de lire l'heure locale de la CPU et d'écrire le résultat dans `DB_HMI_Sync.Date_Time`.
5. La fonction `FC_HMI_Sync` a été appelée depuis `OB1`.
6. Le programme a été téléchargé dans la CPU.

Partie IHM

1. Dans le projet IHM, le menu *Connexions* puis l'onglet *Pointeur de zone* ont été ouverts.
2. La zone globale *Automate date / heure* a été configurée en la reliant à `DB_HMI_Sync.Date_Time`, avec un cycle d'acquisition de `1 min`.
3. Dans l'écran `Convoyeur`, un champ *Date / Heure* en mode sortie a été ajouté et lié à l'heure système du touchpanel via le pointeur de zone.
4. Le projet IHM a été transféré vers le touchpanel et l'affichage a été vérifié.

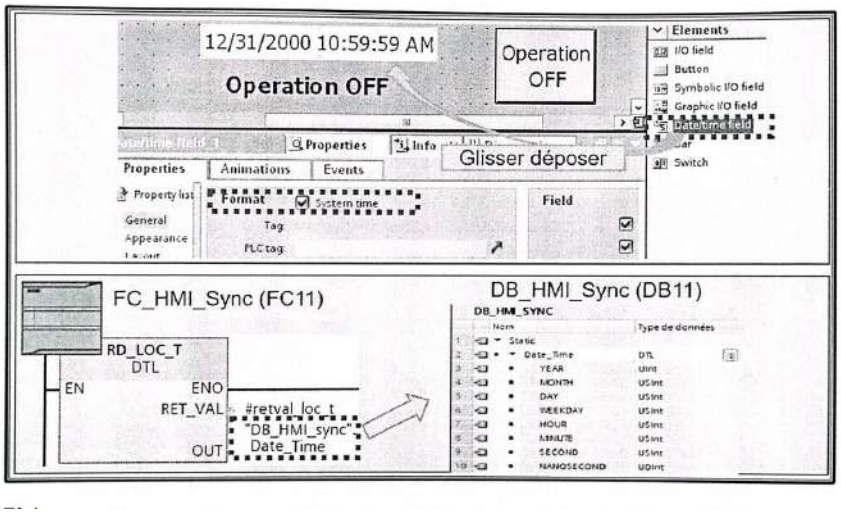


FIGURE 4.26 – Configuration du pointeur de zone `Automate date / heure` et liaison vers `DB_HMI_Sync.Date_Time`

Remarques

— Jusqu'au premier cycle d'acquisition, le touchpanel affiche l'heure système locale ; après la première mise à jour (cycle 1 min), il affichera l'heure de l'automate.

4.8 Objets technologiques

4.8.1 Mise en service d'un contrôleur PID et contrôle d'un moteur pas à pas

Dans un premier temps, un contrôleur PID a été mis en service afin de maintenir la tension aux bornes du condensateur C à une consigne de $10.0V$, même lorsqu'une perturbation (résistance de charge $R3$) était activée via le commutateur S. La variable manipulée (sortie PWM) a été contrôlée par le bloc de contrôleur préfabriqué `PID_Compact`, en évaluant la tension mesurée sur l'entrée analogique A10.

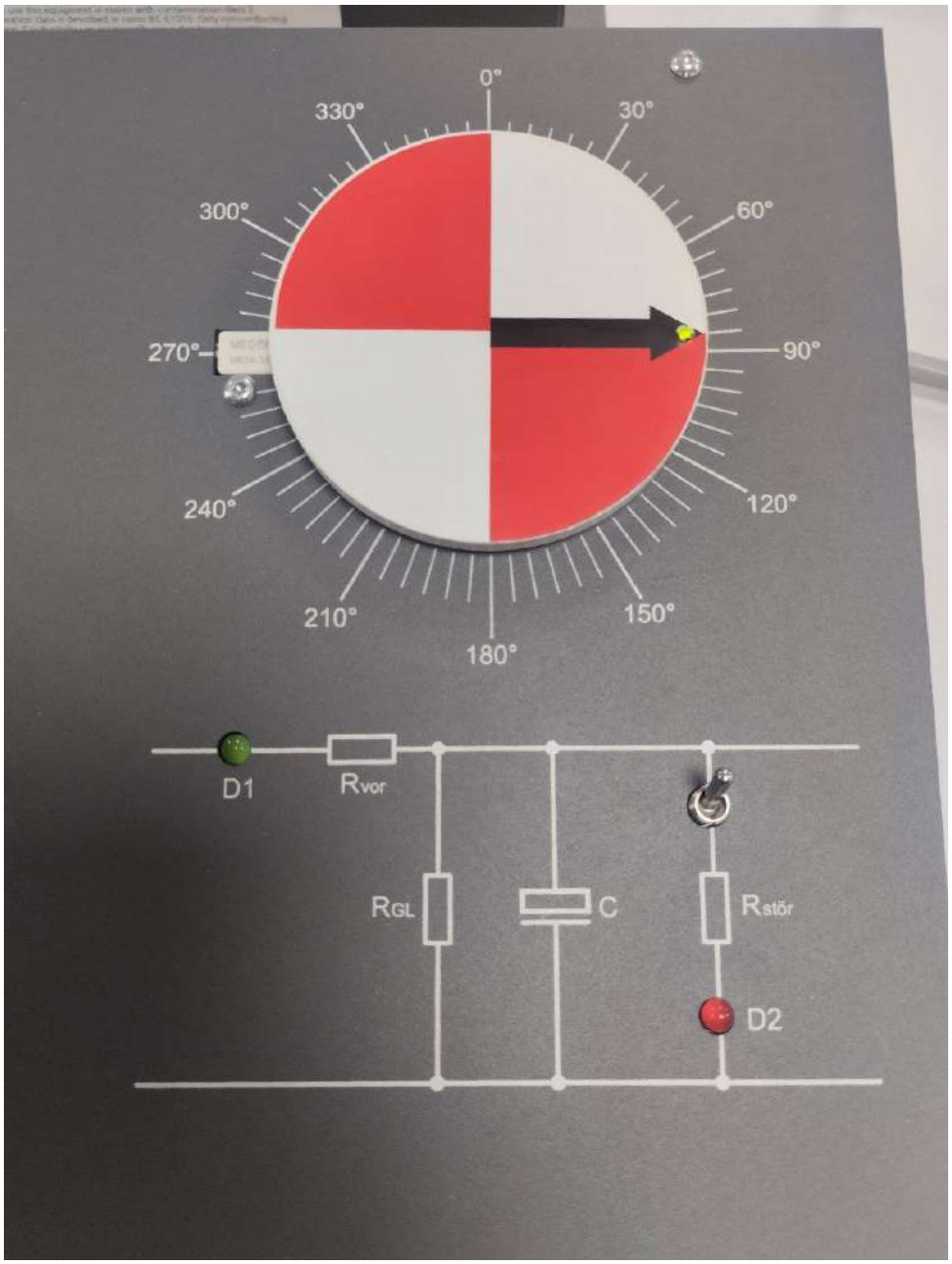


FIGURE 4.27 – Vue d'ensemble : mise en service du PID et de l'axe

Après la mise en service de la boucle PID, le moteur pas à pas de l'appareil de formation a été mis en service à l'aide de l'objet technologique `Axis` (axe de positionnement). La sortie PTO de la CPU et une sortie booléenne pour la direction ont été utilisées. Les séquences de mouvement ont été surveillées via l'interface en ligne (trends / supervision).

4.8.2 Contrôle de la tension du condensateur (PID)

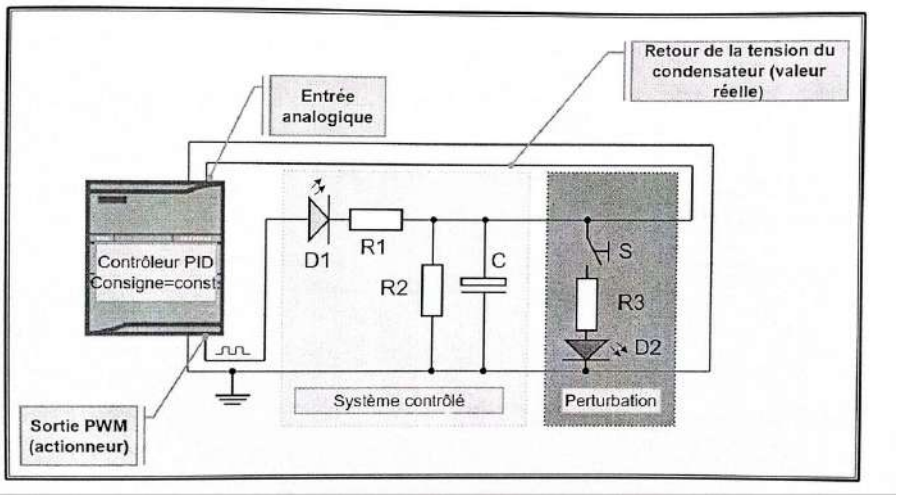


FIGURE 4.28 – Schéma : contrôle de la tension du condensateur par PID

La première étape a consisté à mettre en service la CPU (PLC_2 de type S7-1211C) et à créer l'objet technologique PID :

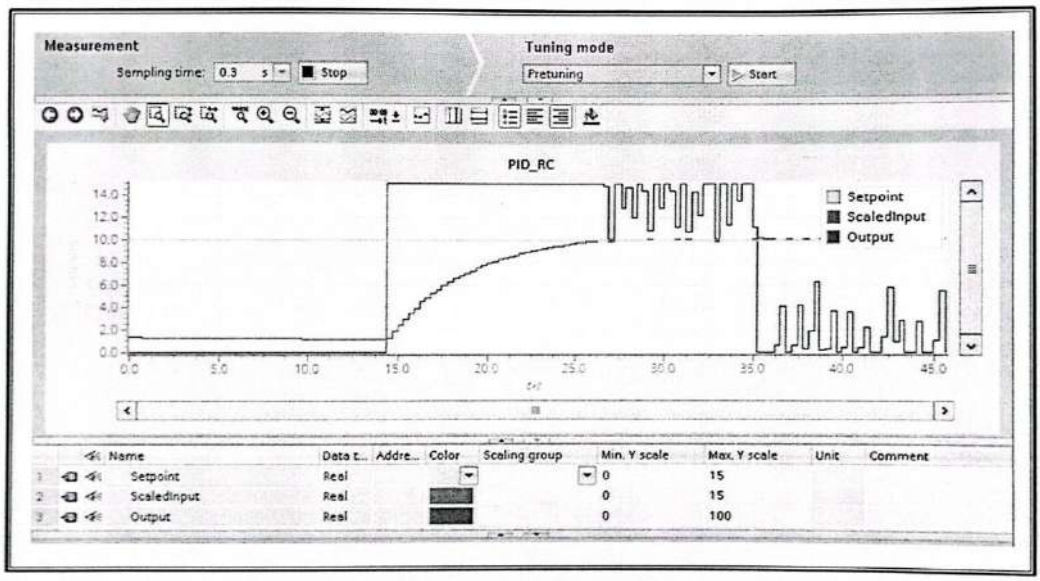
1. Un objet technologique de type `PID` a été créé et nommé `PID_RC`.
2. Une interruption cyclique `OB200` a été ajoutée, dans laquelle le bloc `PID_Compact` a été appelé et lié à l'objet `PID_RC`.
3. Le projet a été enregistré et téléchargé sur la CPU.
4. Le contrôle PID a été mis en service et surveillé en ligne.

Lors des essais :

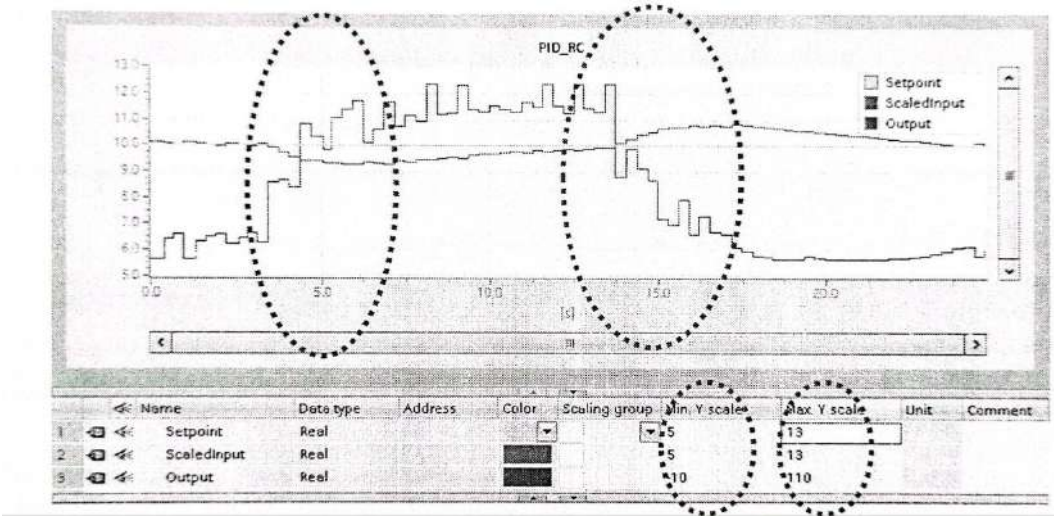
— Les tendances ont été observées (valeur réelle en vert, valeur manipulée en rouge).

— Une perturbation a été activée (résistance $R3$) puis désactivée au bout d'environ 5 s.

— À l'activation de la perturbation, la valeur manipulée est montée immédiatement pour compenser la chute de tension sur le condensateur ; la commande est restée à un niveau élevé (85 %) pendant la perturbation.



(a) Montée de la tension aux bornes du condensateur vers la consigne de 10V



(b) Réaction du régulateur PID lors de l'application d'une perturbation

FIGURE 4.29 – Comportement du régulateur PID : phase de montée et compensation de perturbation

La Figure 4.29 illustre le comportement du régulateur PID. Dans un premier temps, la tension aux bornes du condensateur converge vers la valeur de consigne de 10V. Une fois le régime établi atteint, l'application d'une perturbation provoque une augmentation immédiate de la sortie du régulateur afin de maintenir la tension à la valeur souhaitée.

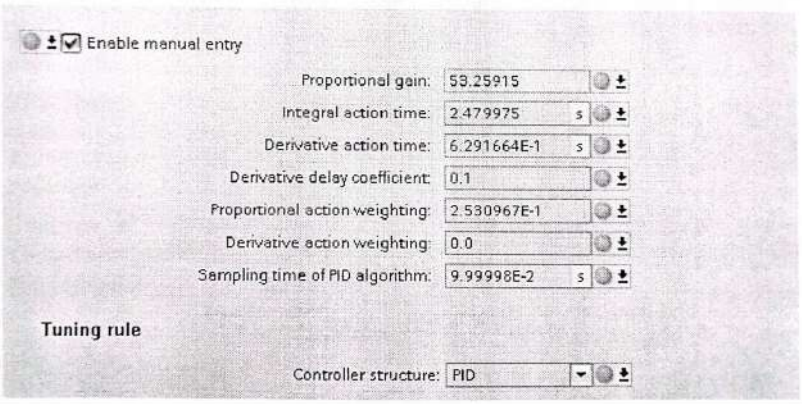


FIGURE 4.30 – Paramètres PID lus par la CPU

4.8.3 Introduction à l'objet technologique `Axis` (contrôle du moteur pas à pas)

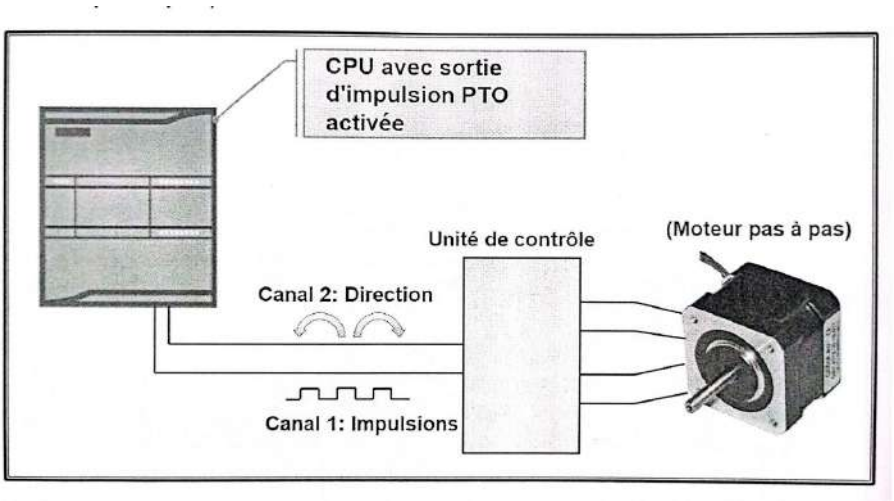


FIGURE 4.31 – Contrôle du moteur pas à pas

L'objet technologique `Axis` a été activé et configuré sur PLC_2. La PTO1 de la CPU a été utilisée pour la commutation pas-à-pas et une sortie booléenne a été dédiée au sens de rotation. Les instructions Motion Control ont été mises à profit pour piloter l'axe.

L'axe a répondu aux commandes absolues/relatives, aux profils de vitesse et aux valeurs d'accélération/décélération configurées ; le mouvement simulé (aiguille rotative) a suivi les ordres avec précision.

4.8.4 Contrôle du moteur pas à pas — Table tournante (`FB_Turntable`)

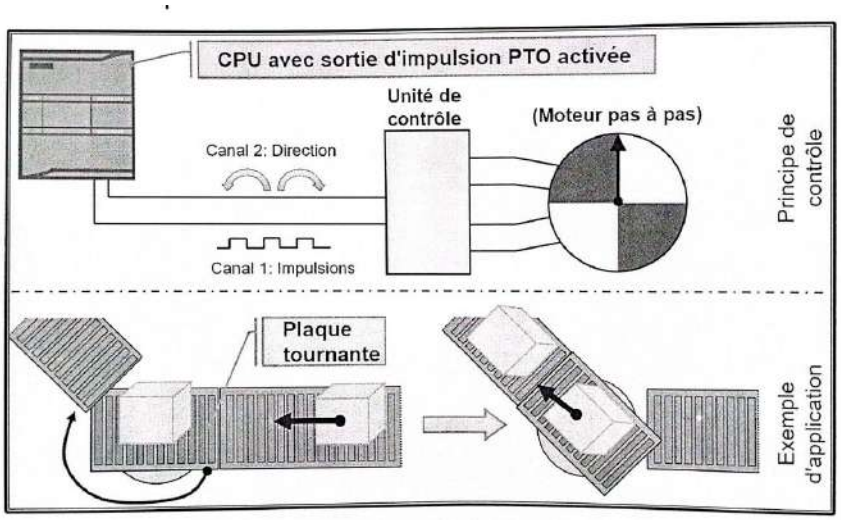


FIGURE 4.32 – Table tournante

Le moteur pas à pas a été piloté via un bloc fonctionnel préfabriqué `FB_Turntable` (FB40). La PTO1 et la sortie de direction ont été affectées à l'axe.

Scénario de fonctionnement

1. Une pièce arrive au poste inférieur et est déposée sur la table (position 1 : 90°).
2. La table de rotation se soulève au niveau supérieur et effectue une rotation de 225° (vers 315° — position 2).
3. La pièce est retirée au niveau supérieur et transportée ; la table retourne ensuite vers la position 1.

Le référencement (homing) a été effectué au démarrage ; le mouvement vertical n'a pas été programmé dans cette exercice. Le code fourni sur clé USB a été intégré, puis `FB_Turntable` a été mis en service.

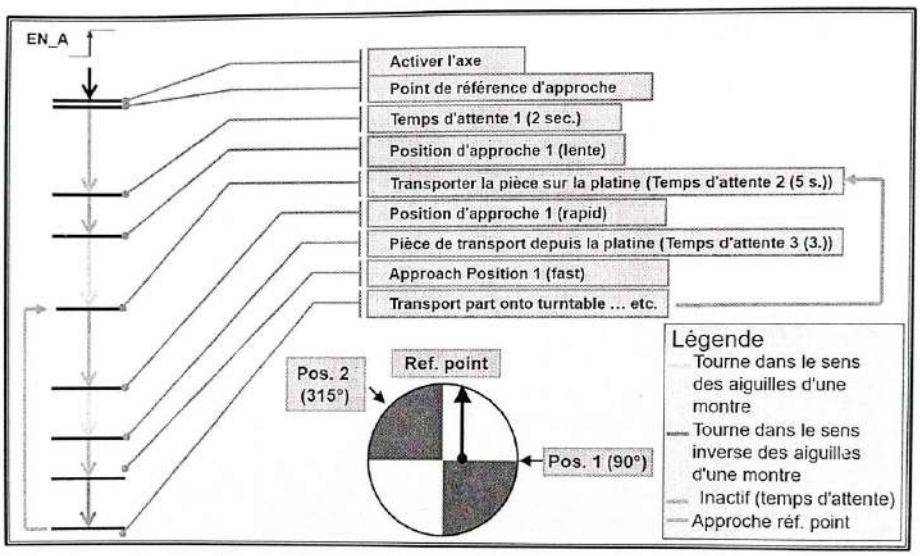


FIGURE 4.33 – Séquence de mouvements supervisée via `FB_Turntable`

Résultat L'axe a d'abord effectué le référencement, puis la séquence de mouvement décrite. Lorsque le bit mémoire `M_Start_Axis` a été remis à `FALSE`, l'axe a terminé l'exécution en cours puis s'est arrêté à la position 1.

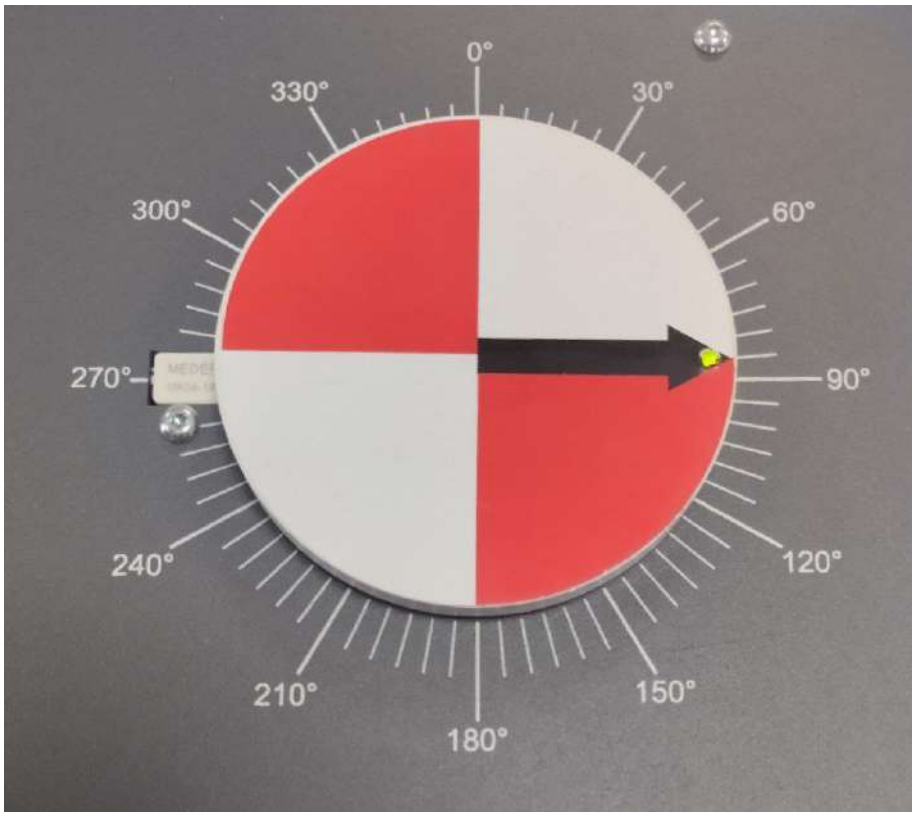


FIGURE 4.34 – Validation : axe à la position 1 après arrêt

4.9 Recherche d'erreurs

Dans le cadre de l'exercice, un programme défaillant fourni par le tuteur a été chargé dans le `PLC_1`. Le formateur nous a remis une clé USB contenant un projet qui réalise exactement les mêmes tâches que notre projet actuel, mais qui comportait volontairement des erreurs à détecter et corriger.

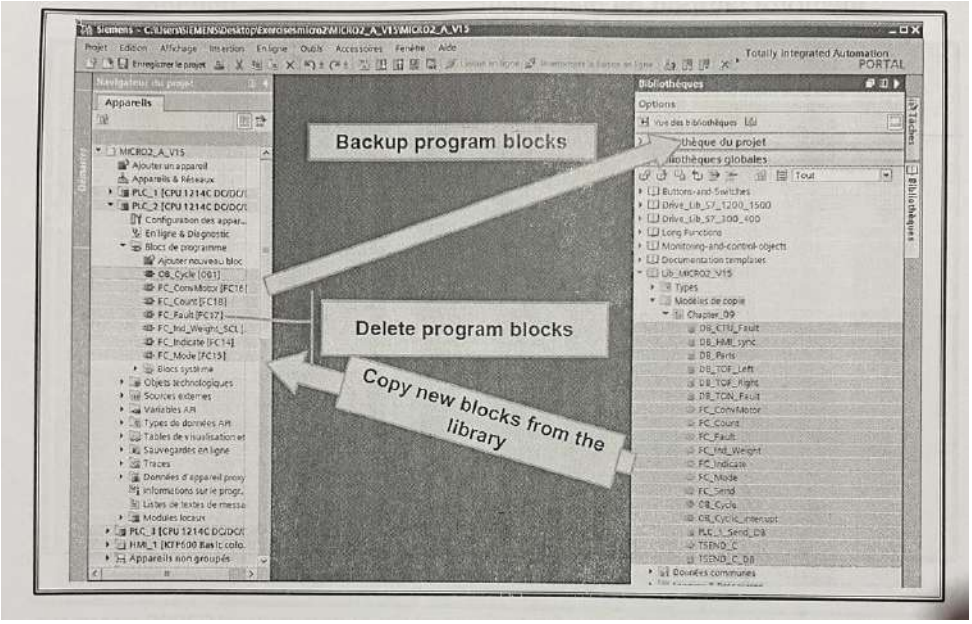


FIGURE 4.35 – Programme défaillant fourni par le tuteur

Les opérations réalisées ont été les suivantes :

1. Sauvegarde de l'ensemble des blocs du programme courant dans la *bibliothèque de projet*, y compris les blocs système, afin de conserver une copie intacte du projet fonctionnel initial.
2. Suppression de tous les blocs du dossier `Program blocks` du projet de travail, puis copie du dossier de blocs livré par le tuteur dans le projet.
3. Compilation du dossier de blocs importés et téléchargement du projet compilé vers la CPU `CPU 1214C` (`PLC_1`).

Remarque de prudence Si le programme de départ est connu comme étant correct, il convient *de ne jamais* le modifier directement. Il faut d'abord en créer une copie et effectuer toutes les modifications sur cette copie. Ainsi, en cas d'erreur, le programme fonctionnel d'origine reste disponible.

4.9.1 Détection et correction des erreurs

La recherche d'erreurs s'est déroulée en deux étapes complémentaires :

Erreurs détectées automatiquement

Nous avons consulté le *tampon de diagnostic* fourni par le système après compilation et téléchargement. Une variable avait été déclarée incorrectement. Nous avons renommé la variable et l'avons déclarée correctement conformément à son usage dans les blocs concernés. Après modification, la compilation n'a plus remonté cette erreur.

Erreurs fonctionnelles (non détectées automatiquement)

Certaines erreurs n'apparaissent pas dans le diagnostic automatique mais provoquent un comportement indésirable à l'exécution. Dans le programme fourni, nous avons identifié une **double affectation** de la même adresse de sortie : `Q8.5`. Ce type d'erreur peut conduire à des conflits et à des états imprévisibles des actionneurs.

La correction a consisté à :

— repérer toutes les occurrences d'affectation sur `Q8.5`,

— redéfinir la logique pour n'avoir qu'un seul point d'écriture effectif,

— recompiler et re-télécharger le projet.

Après ces modifications, des essais en conditions réelles ont été effectués : la logique s'est comportée correctement et le système a retrouvé un fonctionnement conforme aux spécifications.

4.9.2 Bilan

La méthode employée (sauvegarde complète, travail sur copie, usage systématique du tampon de diagnostic puis tests fonctionnels) a permis d'identifier et corriger à la fois des erreurs de déclaration et des erreurs fonctionnelles non visibles au simple contrôle syntaxique. Cette démarche a assuré la remise en service fiable du système sans perte du projet initial.

4.10 SCL

SCL (Structured Control Language) est un langage textuel de haut niveau. Il simplifie la programmation d'algorithmes mathématiques et le traitement de données complexes pour les automates S7. Grâce à SCL, il est possible de réaliser des tâches avancées telles que le contrôle en boucle fermée ou des évaluations statistiques directement dans l'automate.

4.10.1 Fonctionnalités

SCL offre la portée fonctionnelle attendue d'un langage de haut niveau :

— Boucles (`FOR`, `WHILE`, `REPEAT`) ;

— Structures conditionnelles (`IF` / `ELSIF` / `ELSE`) ;

— Gestion des branches et retours (`CASE`, `RETURN`), etc.

Ces constructions sont combinées avec des fonctions spécifiques à l'automate, telles que :

— accès binaires aux E/S et mémoires (bits, mots),

— utilisation des temporisateurs et compteurs,

— accès et manipulation de la table des symboles,

— appels et utilisation de blocs STEP7 (FB, FC, DB).

4.10.2 Avantages de SCL

— Langage relativement facile à apprendre, notamment pour les débutants familiers des langages structurés.

— Code lisible et compact : les algorithmes complexes sont exprimés de façon claire.

— Simplifie la programmation des traitements mathématiques et la manipulation de structures de données (tableaux, enregistrements).

— S'intègre bien avec les autres langages S7 (STL, LAD, FBD) dans un même projet.

— Plus accessible pour des techniciens formés aux langages S7 classiques, grâce à des constructions proches.

4.10.3 Comparaison de LAD et SCL

L'utilisation de SCL est recommandée pour le traitement des données et la programmation de fonctions mathématiques complexes. Dans l'exemple présenté (stockage de données dans une variable tableau en utilisant un adressage indirect), le code SCL est nettement plus compact et lisible que la solution équivalente en langage graphique (LAD / FBD) ou en formes de bloc bas-niveau (CONT / LOG).

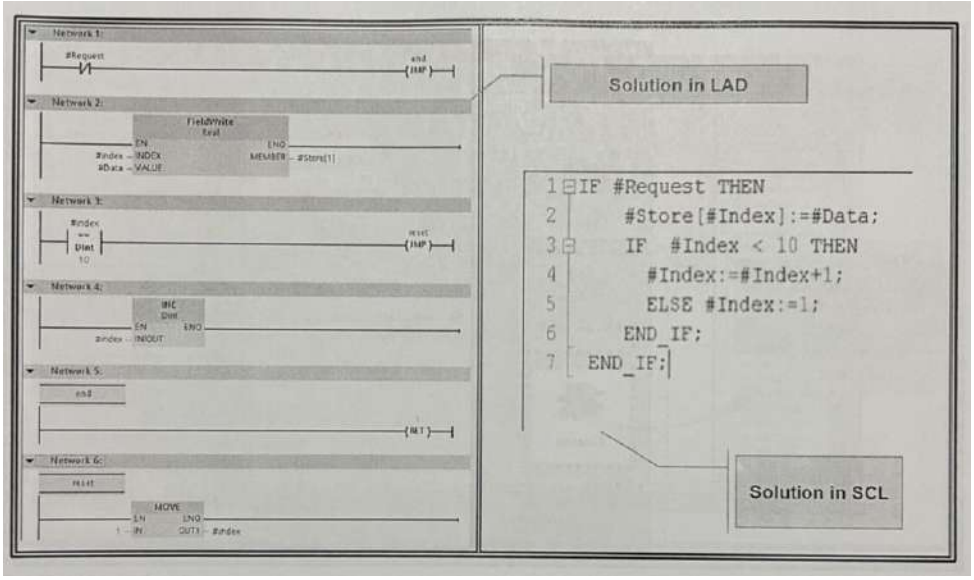


FIGURE 4.36 – Comparaison SCL vs LAD : lisibilité et compacité du code pour le traitement de tableaux.

4.10.4 Conclusion

SCL est particulièrement adapté aux tâches de traitement de données, d'algorithmes numériques et de manipulation avancée de tableaux. Pour des opérations de logique simple et d'interconnexion d'E/S, LAD ou FBD restent pratiques ; pour les algorithmes et le calcul, SCL apporte une grande efficacité de développement et de maintenance.

4.11 Compétences acquises

Liste technique (config matériel, programmation, debug) et non-technique (travail en équipe, rédaction).

Chapitre 5

Formation Maintenace 1

5.1 Présentation générale des gammes S7

Les principales gammes SIMATIC S7 étudiées (classées de la moins performante à la plus performante) ont été présentées : **S7-200**, **S7-1200**, **S7-300**, **S7-1500** et **S7-400**.

Remarque sur le matériel utilisé Nous avons disposé d'un rack S7-300 pouvant accueillir *jusqu'à 11 emplacements* (8 modules). La CPU permettait l'extension par plusieurs racks selon le modèle — le nombre exact d'extensions dépend du modèle de CPU et de la configuration matérielle. Dans notre laboratoire, le matériel utilisé comprenait : une **S7-300**, un **PC portable** avec STEP7, un **simulateur** et une **maquette de bande transporteuse**.

FIGURE 5.1 – Automate SIMATIC S7-300 utilisé lors du stage

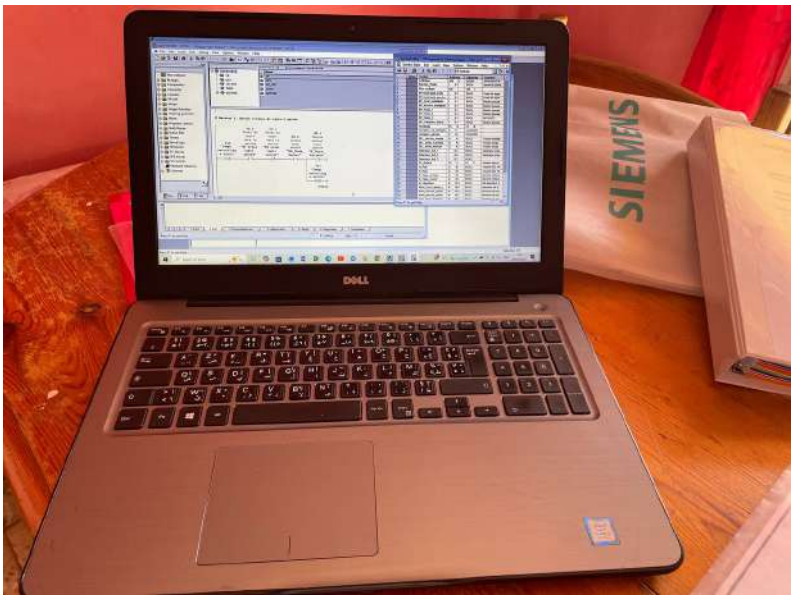


FIGURE 5.2 – PC portable utilisé pour STEP7 / Simatic Manager

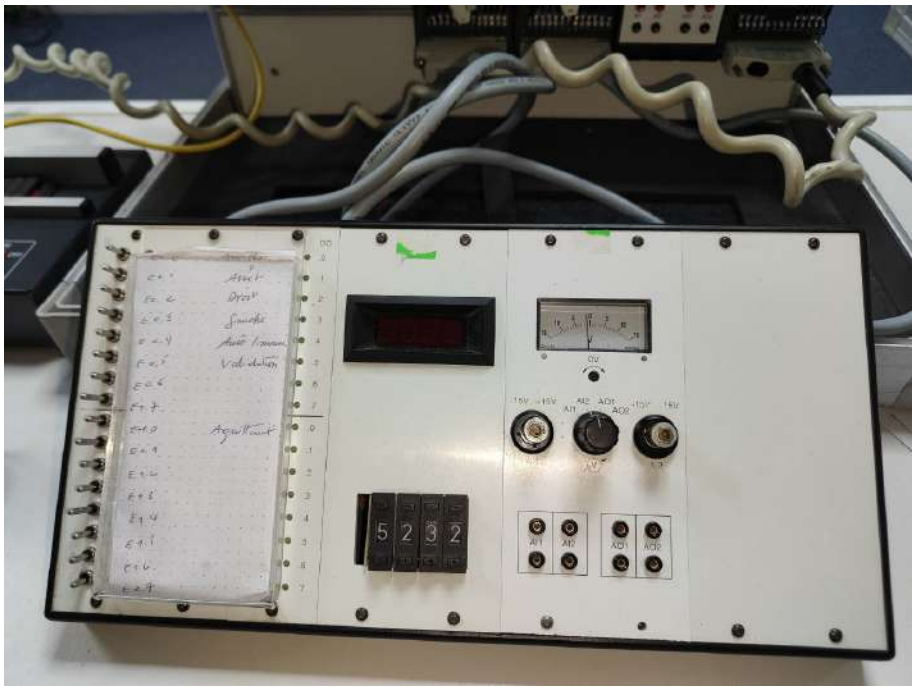


FIGURE 5.3 – Simulateur/Console de formation utilisé pour les essais

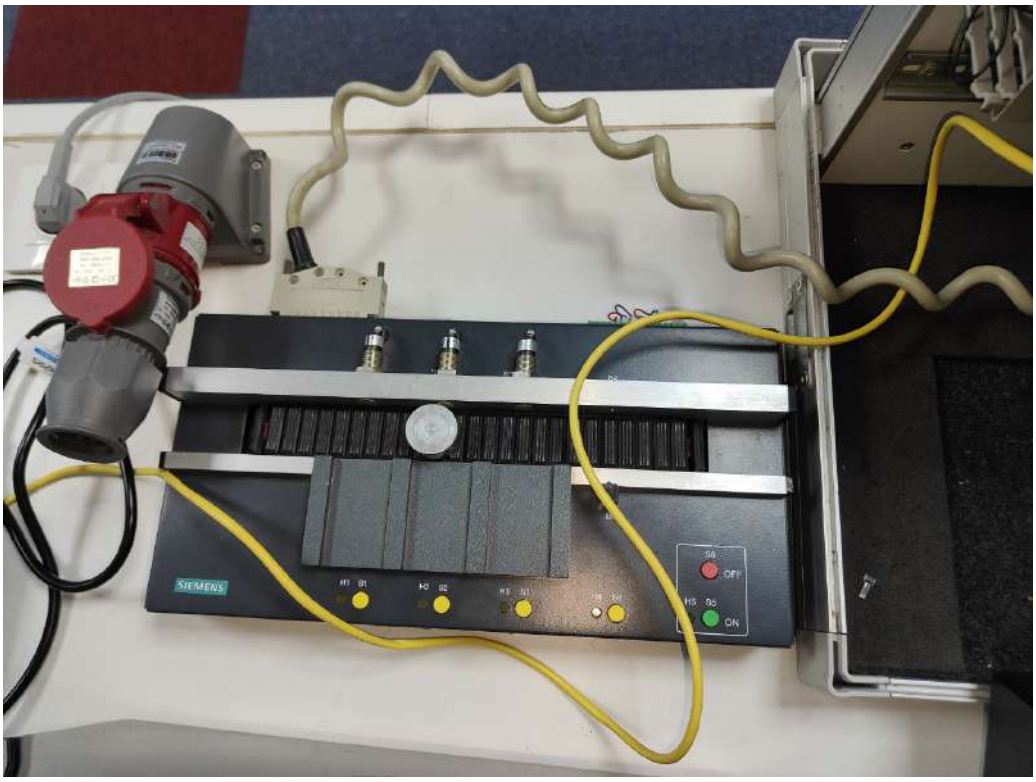


FIGURE 5.4 – Maquette de la bande transporteuse (convoyeur)

5.2 Modules et composants du S7-300

Nous avons revu les types de modules disponibles pour la S7-300 et leur rôle :

— **Modules de signaux (SM)** : modules d'E/S numériques (entrées/sorties) — ex. 24 V DC, relais — pour connecter capteurs et actionneurs.

— **Modules analogiques (AM)** : modules d'entrées/sorties analogiques pour tension/courant (ex. 0–10 V, 4–20 mA).

— **Modules de mesure/température** : modules pour capteurs de température (thermocouples, RTD/resistance).

— **Coupleurs / Interface (IM)** : permettent de relier plusieurs châssis ou réaliser des topologies multi-rack (dans notre cas, le coupleur IM n'était pas présent).

— **Module de réservation (DM)** : emplacement réservé pour un module futur (ex. DM370).

— **Modules de fonction (FM)** : modules fournissant des fonctions spéciales (comptage rapide, positionnement, régulation).

— **Modules de communication (CP)** : permettent les connexions point-à-point, PROFIBUS et Industrial Ethernet.

— **Accessoires** : connecteurs de bus, connecteur frontal, bornes, etc.

5.3 Installation logicielle et premières opérations

Le logiciel **SIMATIC Manager (STEP 7)** a été installé sur le PC et un nouveau projet a été créé.

5.3.1 Effacement et redémarrage de la CPU

Nous avons effectué un effacement général de la CPU puis redémarré l'automate afin de repartir d'un état connu.

5.3.2 Connexion PC–CPU et problème PG/PC

La connexion entre le PC et la CPU a été établie via un câble Ethernet. Nous avons rencontré un problème de communication initial : le PC ne détectait pas la CPU. Le problème a été résolu en modifiant la configuration de l'interface `PG/PC interface` dans STEP7 (sélection de l'interface réseau appropriée).

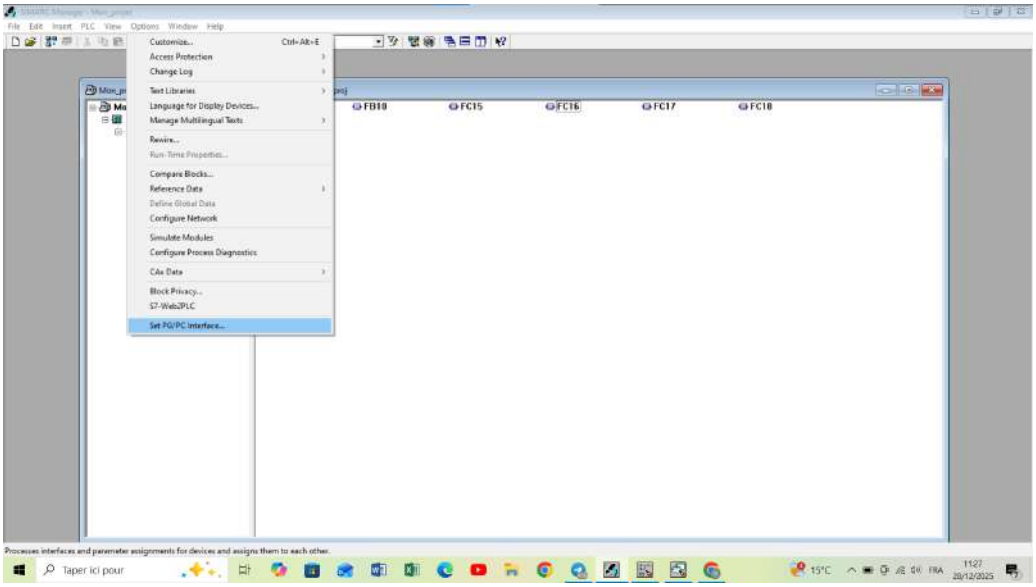


FIGURE 5.5 – Réglage de l'interface PG/PC pour établir la connexion Ethernet avec la CPU

5.3.3 Upload de la station vers PG

À la fin du premier jour, nous avons utilisé l'option *Upload station to PG* pour récupérer la configuration matérielle et les blocs depuis la CPU vers le PC.

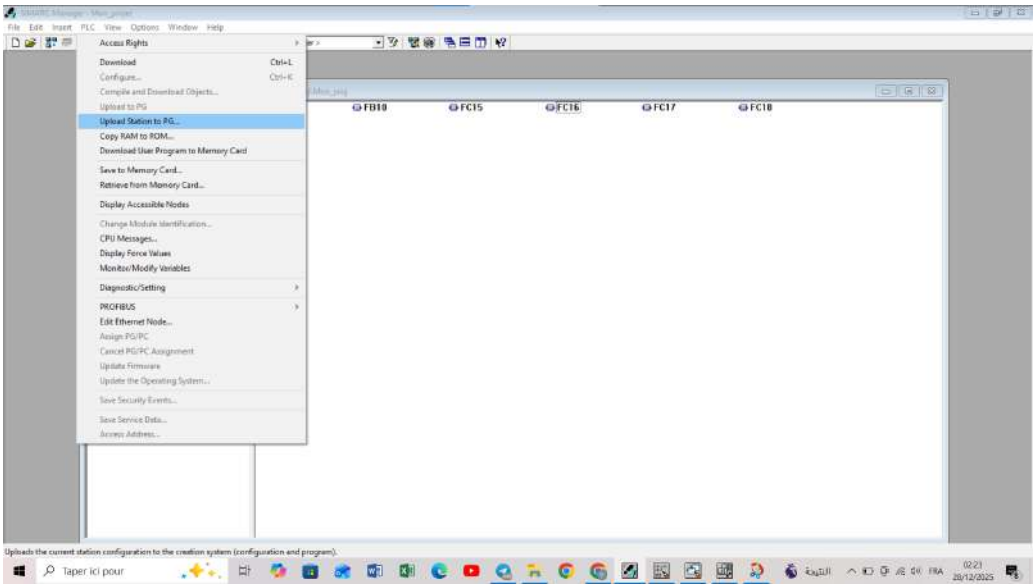


FIGURE 5.6 – Récupération (upload) de la station depuis la CPU vers le PG

5.4 Déclaration des variables et programmation LAD

Le deuxième jour, nous avons déclaré les variables dans la table mémoire (symboles / tags) puis nous avons écrit un programme en langage **LAD (Ladder)** pour piloter des fonctions simples sur le convoyeur : par exemple, en appuyant sur `S1` la lampe `H1` s'allumait et le convoyeur se déplaçait vers la droite ; en appuyant sur `S2` la lampe `H2` s'allumait et le convoyeur se déplaçait vers la gauche.

Symbolique

Editeur de mnémoniques - [Mon programme (Mnémoniques) -- Mon projet]

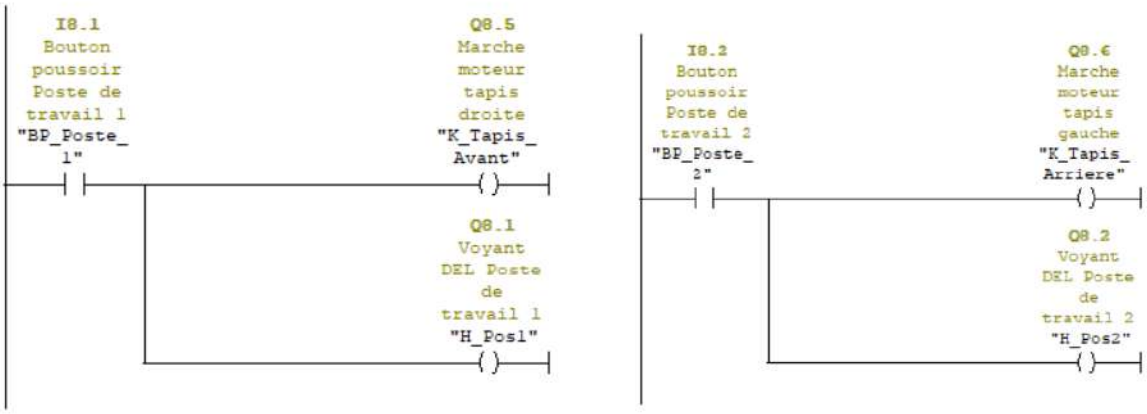
Table Edition Insertion Affichage Outils Fenêtre ?

Tous les mnémoniques

	Etat	Mnémonique	Opérand	Type de don	Commentaire
1		DEL_Defaut_Tapis	A 4.0	BOOL	Voyant defaut tapis
2		DEL_Marche_Installati...	A 4.1	BOOL	Voyant installation en fonctionnement
3		DEL_Mode_Manuel	A 4.2	BOOL	Voyant mode manuel
4		DEL_Mode_Automatiq...	A 4.3	BOOL	Voyant mode automatique
5		DEL_Pos1	A 8.1	BOOL	Voyant DEL Poste de travail 1
6		DEL_Pos2	A 8.2	BOOL	Voyant DEL Poste de travail 2
7		K_Tapis_Avant	A 8.5	BOOL	Moteur tapis vers la droite
8		K_Tapis_Arriere	A 8.6	BOOL	Moteur tapis vers la gauche
9		BP_Marche_installation	E 0.0	BOOL	Bouton poussoir de mise en marche de l'installation
10		BP_Arrêt_Installation	E 0.1	BOOL	Bouton poussoir d'arrêt de l'installation (contact à ouverture)
11		BP_Essai_Tapis_Avant	E 0.2	BOOL	Bouton poussoir d'essai marche à droite du moteur
12		BP_Essai_Tapis_Arri...	E 0.3	BOOL	Bouton poussoir d'essai marche à gauche du moteur
13		S_Présélection Manu/...	E 0.4	BOOL	Commutateur de modes: '0'=MANU, '1'=AUTO
14		BP_Validation_Manu/...	E 0.5	BOOL	Bouton poussoir de validation du mode Auto ou Manuel
15		Cellule	E 8.0	BOOL	Cellule photoélectrique
16		Pos1_BP	E 8.1	BOOL	Bouton poussoir Poste de travail 1
17		Pos2_BP	E 8.2	BOOL	Bouton poussoir Poste de travail 2
18		Det_ind1	E 8.5	BOOL	Détecteur inductif Poste 1
19		Det_ind2	E 8.6	BOOL	Détecteur inductif Poste 2
20		Mode de fonctionnem...	FC 15	FC 15	
21		FC_Commande_Tapis	FC 16	FC 16	
22		FC_Défaut	FC 17	FC 17	Gestion des défauts
23		M_Clignoteur	M 10.3	BOOL	Bit clignoteur 2Hz
24		M_clignteur_1_hz	M 10.5	BOOL	
25		Mem_front_cellule_ph...	M 16.0	BOOL	Memento de front montant de la cellule photo-électrique
26		Mem_marche_droite_...	M 16.2	BOOL	Memoire de demande de fonctionnement a droite du moteur en marche manu
27		Mem_marche_droite_...	M 16.3	BOOL	Memoire de demande de fonctionnement a droite du moteur en marche auto
28					

Pour obtenir de l'aide, appuyez sur F1.

FIGURE 5.7 – Déclaration des variables dans la table mnémonique



(a) Appui sur `S1` : le convoyeur se déplace vers la droite et la LED `H1` est allumée

(b) Appui sur `S2` : le convoyeur se déplace vers la gauche et la LED `H2` est allumée

FIGURE 5.8 – Exécution du programme LAD — commandes S1 / S2 et indication par LED

5.4.1 Bascule RS pour maintien mémoire

Nous avons utilisé une bascule RS (Set/Reset) afin de maintenir l'état mémoire du convoyeur : en appuyant sur `S1` le convoyeur se déplaçait vers la droite et il s'arrêtait lorsque `S3` était activé.

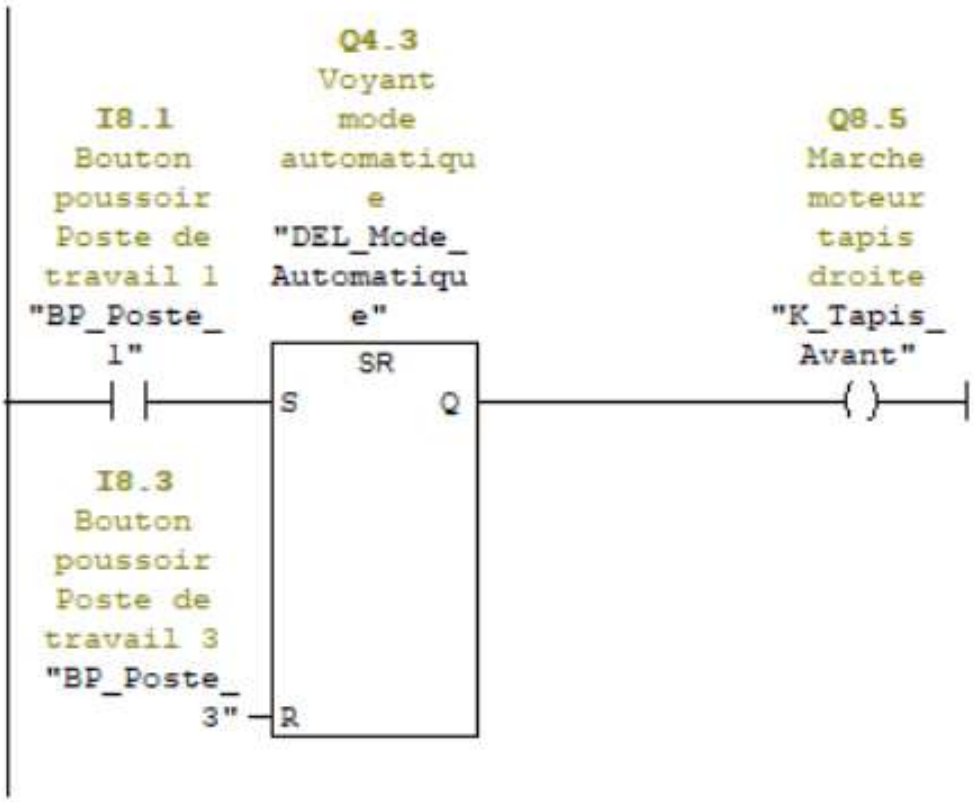


FIGURE 5.9 – Programme LAD : utilisation d'une bascule RS pour mémorisation

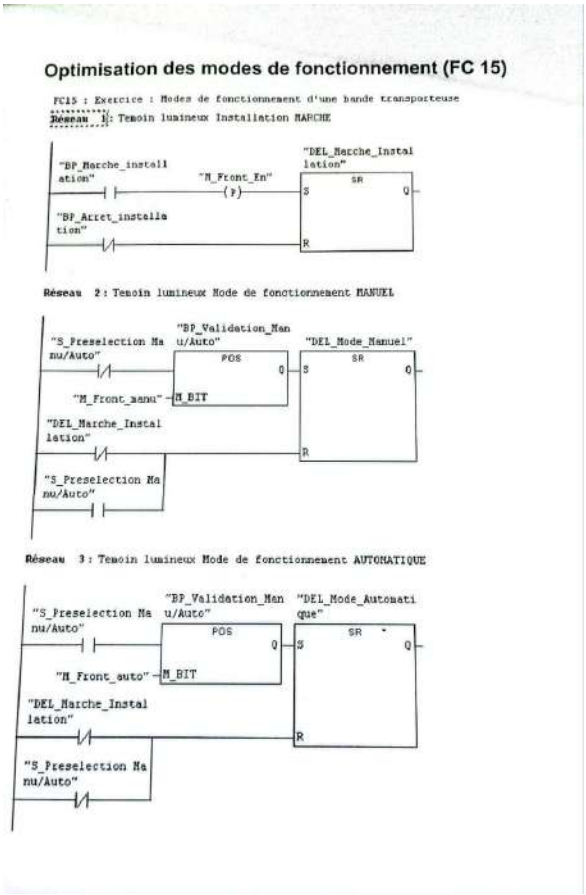
5.5 Import du projet du tuteur et visualisation des blocs

Le troisième jour, le tuteur nous a fourni une clé USB contenant un tableau de variables ainsi que des blocs préconfigurés. Nous avons importé ces éléments, puis appelé les fonctions `FC15`, `FC16` et `FC17` dans l'`OB1`. Après cela, nous avons visualisé leur fonctionnement avant de lancer le convoyeur.

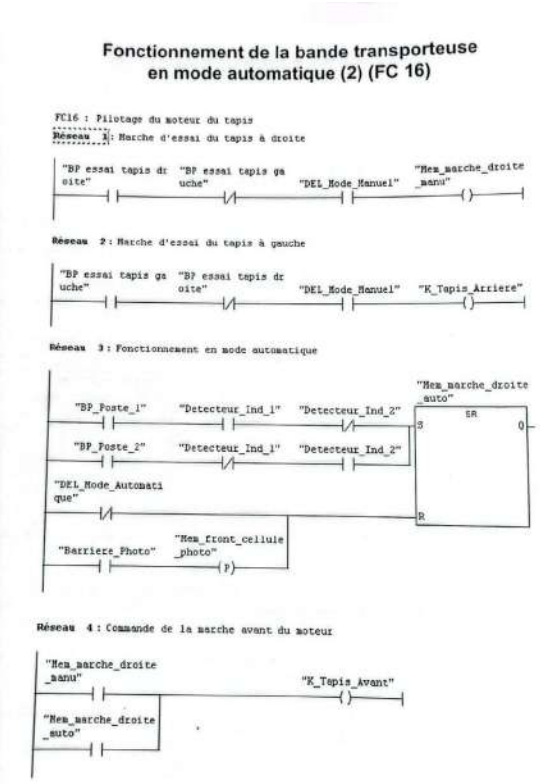
Table des mnémoniques (16 voies)

	Etat	Mnémonique	Opéran /	Type de do	Commentaire
1		H_Défaut_Tapis	A 8.0	BOOL	Voyant défaut tapis
2		DEL_Marche_Installation	A 8.1	BOOL	Voyant installation en fonctionnement
3		DEL_Mode_Manuel	A 8.2	BOOL	Voyant mode manuel
4		DEL_Mode_Automatique	A 8.3	BOOL	Voyant mode automatique
5		DEL_Pos1	A 20.1	BOOL	Voyant DEL Poste de travail 1
6		DEL_Pos2	A 20.2	BOOL	Voyant DEL Poste de travail 2
7		DEL_Pos3	A 20.3	BOOL	Voyant DEL Poste de travail 3
8		DEL_Mes=Cons	A 20.4	BOOL	Voyant Mesure = Consigne
9		K_Tapis_Avant	A 20.5	BOOL	Moteur tapis vers la droite
10		K_Tapis_Arriere	A 20.6	BOOL	Moteur tapis vers la gauche
11		Klaxon	A 20.7	BOOL	Avertisseur sonore
12		Afficheur	AW 12	WORD	Afficheur numérique BCD
13		BP_Marche_Installation	E 0.0	BOOL	Bouton poussoir de mise en marche de l'installation
14		BP_Arrêt_Installation	E 0.1	BOOL	Bouton poussoir d'arrêt de l'installation (contact à ouverture)
15		BP_Essai_Tapis_Avant	E 0.2	BOOL	Bouton poussoir d'essai marche à droite du moteur
16		BP_Essai_Tapis_Arriere	E 0.3	BOOL	Bouton poussoir d'essai marche à gauche du moteur
17		S_Présélection Manu/Auto	E 0.4	BOOL	Commutateur de modes: '0'=MANU, '1'=AUTO
18		BP_Validation_Manu/Auto	E 0.5	BOOL	Bouton poussoir de validation du mode Auto ou Manuel
19		BP_Acquit_Défaut	E 1.0	BOOL	Bouton poussoir d'acquittement des défauts
20		Cellule	E 16.0	BOOL	Cellule photoélectrique
21		Pos1_BP	E 16.1	BOOL	Bouton poussoir Poste de travail 1
22		Pos2_BP	E 16.2	BOOL	Bouton poussoir Poste de travail 2
23		Pos3_BP	E 16.3	BOOL	Bouton poussoir Poste de travail 3
24		Pos_Fin_BP	E 16.4	BOOL	Bouton poussoir Poste final
25		Det_ind1	E 16.5	BOOL	Détecteur inductif Poste 1
26		Det_ind2	E 16.6	BOOL	Détecteur inductif Poste 2
27		Det_ind3	E 16.7	BOOL	Détecteur inductif Poste 3
28		Roue_codeuse	EW 4	WORD	Rouses codeuses BCD
29		Mode de fonctionnement	FC 15	FC 15	
30		FC_Commande_Tapis	FC 16	FC 16	
31		FC_Défaut	FC 17	FC 17	Gestion des défauts
32		FC_Traitement_Données	FC 18	FC 18	Comptage des pièces, comparaison mesure=consigne
33		M_Clignoteur	M 10.3	BOOL	Bit clignoteur 2Hz
34		M_clignoteur_1_hz	M 10.5	BOOL	
35		front_desc_compteur	M 15.7	BOOL	
36		Mem_front_cellule_photo	M 16.0	BOOL	Memento de front montant de la cellule photo-electrique
37		Mem_front_compteur	M 16.1	BOOL	Memento de front montant du compteur
38		Mem_marche_droite_manu	M 16.2	BOOL	Memoire de demande de fonctionnement a droite du moteur en marche manu
39		Mem_marche_droite_auto	M 16.3	BOOL	Memoire de demande de fonctionnement a droite du moteur en marche auto
40		Mem_Défaut_Tapis	M 17.0	BOOL	Mémento Défaut Tapis en automatique
41		MW_Nombre_de_pieces	MW 20	WORD	Mot nombre de pieces
42		Tempo verrouillage droit	T 15	TIMER	Tempo verrouillage en manu droite
43		Tempo verrouillage gauc	T 16	TIMER	Tempo verrouillage en manu gauch
44		compteur_nb_defauts	Z 17	COUNTER	Compteur limitant le mode auto au nombre de 3 defauts
45		compteur_pieces	Z 18	COUNTER	Compteur pieces
46					

FIGURE 5.10 – Projet fourni par le tuteur sur clé USB (tableau de variables)



(a) Code LAD — FC15

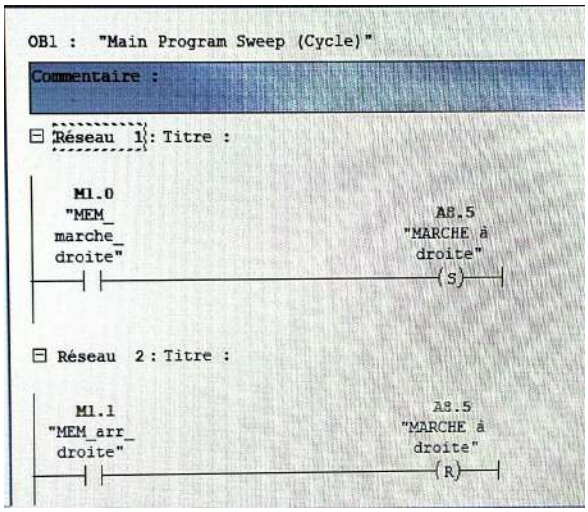


(b) Code LAD — FC16

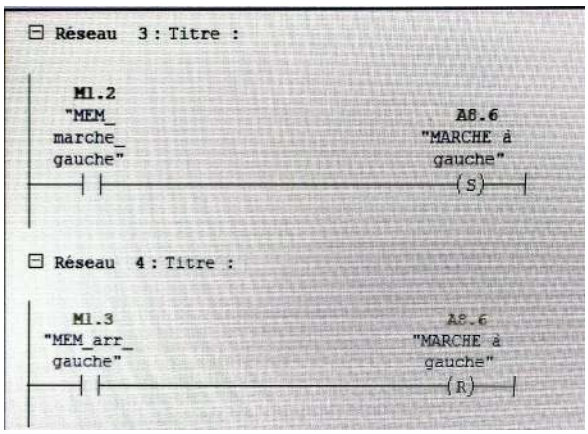
FIGURE 5.11 – Extraits LAD : FC15 et FC16 (visualisation et test sur la maquette)

5.6 Séquence pièce et résolution d'un problème de logique

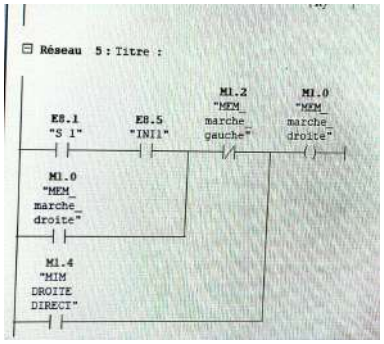
Le quatrième jour, nous avons programmé une séquence suivante : une pièce devait partir de `S1` vers `S3`, effectuer une pause de 5 s, revenir vers `S2` et faire une pause de 10 s, puis partir vers `S4` où elle était comptée. Lors des tests, la pièce s'arrêtait à `S3` lors du trajet de `S2` vers `S4` : la condition qui l'empêchait de continuer restait présente. Nous avons résolu ce problème en ajoutant une mémoire (bit intermédiaire) qui s'activait lorsque la pièce s'arrêtait à `S2` ; cette mémoire a permis de supprimer la condition bloquante et d'autoriser la poursuite correcte de la séquence.



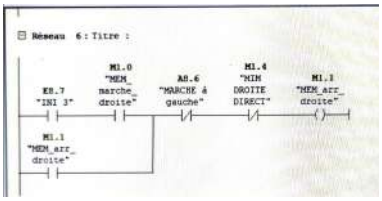
(a) Réseaux 1 & 2 (extrait combiné)



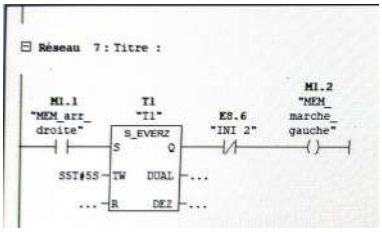
(b) Réseaux 3 & 4 (extrait combiné)



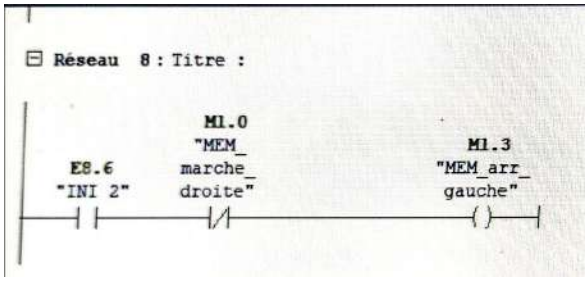
(c) Réseau 5



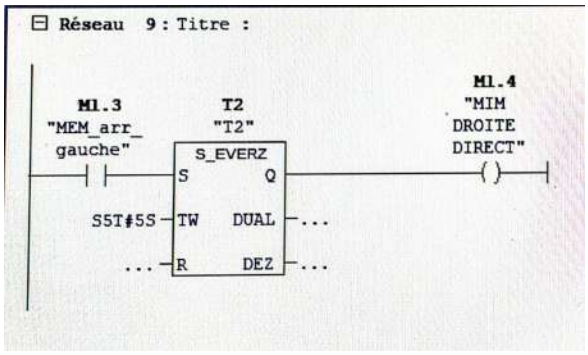
(d) Réseau 6



(e) Réseau 7



(f) Réseau 8



(g) Réseau 9

FIGURE 5.12 – Extraits du programme LAD par réseaux (1&2 et 3&4 combinés, suivis des réseaux 5 à 9)

5.7 Bilan et remarques

— La prise en main de STEP7 (Simatic Manager) et la configuration matérielle ont permis de comprendre le flux complet : déclaration de variables, écriture LAD, compilation, téléchargement et tests sur la maquette.

— La méthode de sauvegarde et d'upload des stations a été utile pour travailler en sécurité sur des copies du projet.

— La résolution d'erreurs logiques (séquence bloquante) a illustré l'importance des mémoires intermédiaires et du phasage correct des conditions dans les applications séquentielles.

Chapitre 6

Mobility (métro)

6.1 Introduction

Durant la formation, un ensemble de notions liées aux projets de mobilité urbaine et ferroviaire nous a été présenté : objectifs environnementaux (Siemens *« green mobility »*), modèles commerciaux (projet *clé en main*), architecture des systèmes métro, modes d'alimentation électrique, commandes automatiques et sûreté. La section suivante synthétise ces notions, complète les points abordés par des explications techniques et fournit le vocabulaire utile pour un rapport technique.

6.2 Projets *clé en main* et partenariats

6.2.1 Concept

Un projet dit *clé en main* signifie que le client contracte principalement avec un intégrateur (ici : Siemens) qui assure la responsabilité globale — conception, fourniture, installation, tests et mise en service. Le client a donc un seul interlocuteur commercial même si l'intégrateur sous-traite des lots spécialisés.

6.2.2 Mode de coopération industrielle

Siemens peut faire appel à des partenaires industriels (constructeurs de matériel roulant, fournisseurs d'armement, d'infrastructure) — par exemple CAF, Alstom, Bombardier/Siemens Mobility, etc. — afin de fournir des sous-ensembles (rames, automates de traction, signalisation, matériel d'infrastructure). Le client final connaît souvent uniquement l'intégrateur principal ; la collaboration entre fournisseurs demeure cependant essentielle pour garantir l'intégration système.

6.3 Alimentation électrique et modes de traction

6.3.1 Types d'alimentation

Les métros et trains peuvent être alimentés de deux manières principales :

— **Caténaire / pantographe** : ligne aérienne et captation par pantographe — utilisée sur lignes ferroviaires et pour de nombreuses lignes de métro modernisées.

— **Troisième rail** : troisième rail latéral ou central alimentant les véhicules via un frotteur — fréquemment employé sur des réseaux métro urbains à faible gabarit.

6.3.2 Régénération et freinage négatif

Le *freinage régénératif* (parfois appelé *freinage négatif*) désigne l'utilisation du moteur en tant que générateur : lors du freinage, l'énergie cinétique du véhicule est convertie en énergie électrique et renvoyée sur le réseau (ou dissipée si le réseau ne peut pas l'absorber). Cela améliore l'efficacité énergétique et réduit l'usure mécanique.

6.4 Signalisation et contrôle automatique

6.4.1 CBTC — Communication-Based Train Control

Le CBTC est un système de contrôle et de protection du trafic fondé sur l'échange continu d'informations entre les trains et l'infrastructure via radio. Ses caractéristiques clés :

— positionnement continu (dense traffic headway),

— supervision en temps réel (vitesse et séparation),

— possibilité d'exploitation automatique (various GoA levels).

CBTC permet d'augmenter la capacité d'une ligne et de réduire l'intervalle entre trains.

6.4.2 Niveaux d'automatisation (GoA)

La norme distingue plusieurs grades d'automatisation (GoA) :

GoA 0 : conduite manuelle sans protection automatique,

GoA 1 : conduite manuelle avec ATP (Automatic Train Protection),

GoA 2 : conduite semi-automatique (ATO supervise les départs et arrêts ; conducteur vigilant),

GoA 3 : conduite automatique sans conducteur à bord (attended-less),

GoA 4 : conduite totalement automatique (UTO — unattended train operation).

Le VAL (Véhicule Automatique Léger) est un exemple de système conçu pour GoA 4 sur certains réseaux (voies pneumatiques / pneus en caoutchouc, exploitation sans conducteur).

6.5 PCC / Centre de Contrôle et supervision

6.5.1 Rôle du PCC

Le *Poste Central de Commandement* (PCC) ou *Operations Control Center* supervise l'exploitation : gestion des circulations, incidents, aiguillages, horaires, interface avec le CBTC, gestion des dépôts et des plans de service. Siemens peut livrer la partie logicielle de PCC et intégrer l'interface HMI/SCADA correspondante.

6.5.2 Fonctions critiques

— surveillance en temps réel des trains et des signalisations,

— remontée d'alarmes et actions d'exploitation,

— gestion des plans de transport et des priorités (dépannage, régulation).

6.6 Sûreté et dispositifs embarqués

6.6.1 Dispositif d'alerte conducteur (*homme mort* / vigilance)

La *dispositif de vigilance* ou *homme mort* oblige le conducteur à maintenir une action périodique (pédale, bouton). À défaut, une alarme sonore est générée (ex. 3 bips) puis, si aucune réaction, un freinage d'urgence est déclenché pour arrêter le train en sécurité.

6.6.2 Zonage électrique et coupures d'urgence

Les installations électriques sont segmentées en zones de puissance ; une coupure ou un rupteur de secours sur une zone déclenche une alerte locale et centrale et peut isoler la zone pour protéger le matériel ou le personnel.

6.7 Localisation des trains et gestion du trafic

6.7.1 Techniques de localisation

Pour connaître la position exacte d'un train, on combine souvent plusieurs sources :

— odométrie embarquée (odomètre, capteurs d'axe),

— balises *beacons* au sol (balises RFID, inductives) pour recalage,
— échanges radio CBTC (position estimée partagée en continu),
— systèmes GNSS (rare en tunnels, utile en sections ouvertes).

6.7.2 Prévention des collisions et régulation

La gestion automatique des trains (via CBTC/ATO et ATP) garantit la séparation minimale entre trains, ajuste les vitesses et priorise les mouvements afin d'éviter les collisions et d'optimiser la capacité.

6.8 Réseaux multiservices (RMC) et infrastructure de communication

6.8.1 Réseau multiservices

Le *Réseau Multiservices* (RMC) supporte plusieurs classes de trafic : voix, données SCADA/PCC, téléphonie, vidéosurveillance et CBTC. Sa conception priorise la redondance, la QoS (quality of service) et la résilience.

6.8.2 Infrastructure physique

L'infrastructure ferroviaire comprend : voies, ballast ou dalle, traverse, caténaires/3^{e} rail, stations, tunnels, systèmes de drainage, alimentation stationnaire et postes de traction.

6.9 Contexte algérien et projets évoqués

6.9.1 Projets mentionnés pendant la formation

Il nous a été indiqué qu'un projet de voie partant de *Gare Jebilet* vers Oran était en préparation et que des locomotives de type *VECTRON* seraient fournies à la SNTF (exemple cité : 30 unités). Ces informations proviennent de la présentation reçue ; pour un rapport officiel, il conviendrait de confirmer les chiffres et contrats via des sources officielles.

6.9.2 Réseaux ferrés et électrification en Algérie

On a évoqué la configuration des lignes (électrifiées ou non) et le rôle de l'électrification pour la performance et la durabilité : la traction électrique permet la régénération, la réduction des émissions locales et des coûts d'exploitation.

6.10 Conclusion — enjeux et perspectives

Les systèmes de mobilité moderne reposent sur l'intégration étroite de l'infrastructure, du matériel roulant, des systèmes de signalisation (CBTC) et des centres de supervision (PCC). Les tendances actuelles mettent l'accent sur :

— l'efficacité énergétique (récupération d'énergie, optimisation des trajectoires),

— l'automatisation (augmentation du GoA pour plus de capacité),

— la cybersécurité et la résilience des réseaux de communication,

— la prise en compte environnementale (matériaux, bruit, empreinte carbone).

Chapitre 7

Analyse critique et bilan

7.1 Apports du stage

Ce stage a représenté une expérience particulièrement enrichissante sur plusieurs plans :

Sur le plan technique :

— Acquisition d'une solide maîtrise des environnements TIA Portal et STEP7

— Compréhension approfondie des architectures réseaux industriels (PROFINET, communication TCP/IP)

— Développement de compétences en programmation structurée et gestion de données

— Maîtrise des techniques de diagnostic et débogage de programmes automates

— Expérience concrète sur équipements industriels actuels

Sur le plan méthodologique :

— Approche systématique de résolution de problèmes techniques

— Développement d'une rigueur dans la documentation et la structuration des projets

— Pratique du travail en équipe avec des professionnels expérimentés

— Adaptation à des environnements techniques complexes et évolutifs

Sur le plan professionnel :

— Immersion dans la culture et les standards d'un leader mondial de l'automatisation

— Compréhension des enjeux industriels réels et des contraintes de production

— Élargissement du réseau professionnel avec des experts et des pairs du secteur

Chapitre 8

Conclusion

Ce stage de formation chez Siemens a constitué une étape déterminante dans mon parcours d'ingénieur en automatique. Au-delà de l'acquisition de compétences techniques pointues sur les automates SIMATIC, il m'a offert une immersion concrète dans les pratiques et standards d'un leader mondial de l'automatisation industrielle.

Les deux formations complémentaires - TIA-MICRO2 et Maintenance 1 - ont couvert un spectre large et cohérent, depuis la programmation de base jusqu'aux architectures réseaux avancées. L'alternance entre théorie et pratique sur équipements réels a été particulièrement efficace pour ancrer les connaissances et développer une approche méthodologique.

Les échanges avec les formateurs experts et les autres stagiaires issus du milieu industriel (Sonatrach, Naftal) ont également été très enrichissants, offrant des perspectives concrètes sur les applications réelles et les enjeux du terrain.

Je retiens de cette expérience l'importance de la rigueur dans la programmation, la nécessité d'une documentation précise et la complexité croissante des systèmes automatisés modernes, qui nécessitent des compétences à la fois techniques et systémiques.

Ce stage a renforcé ma motivation à poursuivre dans le domaine de l'automatisation industrielle et m'a donné les bases solides pour aborder sereinement les défis techniques des projets industriels futurs. Je remercie sincèrement Siemens pour la qualité de l'accueil et de la formation dispensée, ainsi que l'École Nationale Polytechnique pour avoir facilité cette opportunité.