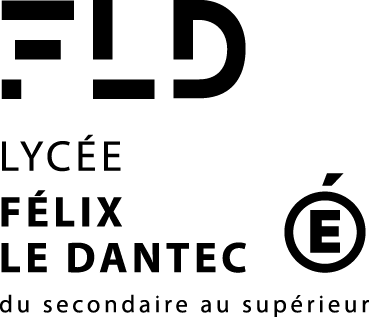
**BTS**

**Systèmes Numériques**

**Option Informatique et Réseaux**

**Session 2024**

**Lycée Félix Le Dantec- Lannion**

**Chaine de manutention**

\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*

**Rapport de projet**

Partie individuelle

[**1. Introduction au projet 4**](#_uoga39sl5u6t)

[1.1 Objectifs à atteindre 4](#_yiujej1qhkme)

[1.1.1 objectifs 4](#_8bs98of9iaey)

[1.1.2 Objectifs de Performance 5](#_tk2vb41v4sue)

[1.2 Répartition des tâches 6](#_z75mq4aei2j0)

[1.3 cahier des charges personnelle 9](#_s1nophnrrr4a)

[1.4 Gestion de projet 10](#_r6etdem401pi)

[**2. Situation dans le projet 11**](#_qdm65h1s0wpr)

[2.1 Analyse des besoins 11](#_pn28f6aga6cp)

[2.2 – Analyse des exigences 11](#_tv2s1828bo48)

[2.2.1 Interface Homme-Machine (IHM) : 11](#_pihdhvcaxtww)

[2.2.2 Contrôle du Tapis Roulant : 12](#_bx7elulquxtb)

[2.2.3 Afficheur Industriel : 12](#_gitlegxn9q66)

[2.2.4 Interface Technicien : 12](#_oft4rvhp1yjv)

[**3. Réalisation 14**](#_sw2ijgyiloke)

[3.1 Configuration 14](#_q5djimm6mhc2)

[3.2 Matériels mis à disposition 15](#_9hi70fdt6pub)

[3.3 Câblage Boîtier E/S 18](#_1ewu91ct34cx)

[3.4 logiciels et langages de programmation 20](#_6hz39su865tz)

[3.5 Contrôle-Commande du Convoyeur 22](#_dh55h8ioyahm)

[3.5.1 Communication 22](#_n02gxgvw9cjj)

[3.5.2 Fonctionnalité 22](#_owdhbiqrclvy)

[3.5.3 Mode automatique 23](#_4z3q7bi45ly2)

[3.5.4 Partie Programmation 26](#_d8lyav8kaplh)

[3.6 Afficheur industrielle 31](#_nohdppuntyrf)

[3.7 IHM Globale 31](#_9r7zpyx7c1rc)

[3.8 Tests unitaires 32](#_mta7alruhwhs)

[**4. Bilan des travaux 32**](#_lvcwg5omrbry)

[**5. Annexes 34**](#_r40j2myfw69)

**Introduction**

Le sujet de notre projet est une chaîne de manutention robotisée. C'est un projet qu'on fait au lycée Félix Le Dantec à Lannion pour notre formation en BTS Systèmes Numériques.

On va travailler sur deux robots : un FANUC Mi10 ou Mi20, et un robot didactique Niryo Ned2. Le premier robot est utilisé pour prendre des pièces dans un bac et les poser sur un tapis roulant. Le second robot prend les pièces du tapis et les met dans un autre bac. Tout le système sera contrôlé par une application graphique qui supervise les opérations en utilisant le protocole Modbus/TCP.

L'objectif est de créer une solution automatisée pour déplacer des pièces entre deux postes de stockage, avec une supervision globale et des commandes précises pour chaque robot et le tapis roulant.

# [1. Introduction au projet](#_wxb1rp8qqwmq)

## 

## 1.1 [Objectifs à atteindre](#_wgucpj92s0nz)

Le projet a pour objectif d'automatiser une chaîne de manutention en utilisant deux robots distincts.

### 1.1.1 objectifs

**Robot FANUC Mi10 ou Mi20 :**

Installation et Sécurisation : Le robot FANUC, un robot industriel robuste et polyvalent, est installé dans notre atelier du lycée Félix Le Dantec. Il est sécurisé par une enceinte grillagée pour garantir la sécurité des opérateurs et des étudiants lors de son fonctionnement.

Fonctionnement Principal : Ce robot est responsable de prendre des pièces depuis le "Poste de stockage 1". Lorsqu'il détecte que le compteur de pièces (CptPiece1) est à zéro, il prend une pièce du bac et la place sur le tapis roulant.

**Tapis Roulant (Convoyeur):**

Transport des pièces : Le tapis roulant motorisé transporte les pièces du "Poste de stockage 1" au "Poste de stockage 2". Il est équipé de capteurs aux deux extrémités (CptPiece1 et CptPiece2) pour détecter la présence des pièces.

Contrôle et Commande : Le mouvement du tapis est contrôlé pour avancer jusqu’à ce que le compteur de pièces du "Poste de stockage 2" (CptPiece2) indique la présence d'une pièce.

**Robot Didactique Niryo Ned2 :**

Contrôle via Écran Tactile: Ce robot est contrôlé par un écran tactile utilisant le protocole Modbus/TCP, ce qui permet une interface utilisateur intuitive et facile à manipuler.

Fonctionnement Principal : Une fois que le tapis roulant a déplacé une pièce jusqu’au "Poste de stockage 2", le robot Niryo la prend du tapis roulant et la dépose dans le bac correspondant.

**Synchronisation et Supervision :**

Supervision Globale : Supervision de l'ensemble du système automatisé, assurant la synchronisation parfaite entre le robot FANUC, le tapis roulant, et le robot Niryo.

Interface Homme-Machine (IHM) : Développement d'une application graphique pour initialiser les matériels, démarrer et arrêter le cycle de production, synchroniser les éléments du système, et afficher l'état de production en temps réel.

Protocole de Communication : Utilisation du protocole Modbus/TCP pour permettre la communication entre les différents éléments de la chaîne de manutention, garantissant une coordination fluide et efficace.

### 1.1.2 Objectifs de Performance

Efficacité : Minimiser les temps d'arrêt et maximiser la vitesse de manutention.

Fiabilité: Assurer un fonctionnement sans erreur, avec des capteurs et des robots parfaitement synchronisés.

Sécurité : Maintenir des normes de sécurité élevées pour éviter tout risque pour les utilisateurs.

Facilité d'Utilisation : Développer une interface utilisateur intuitive pour faciliter le contrôle et la supervision du système.

Pour conclure le projet de chaîne de manutention robotisée vise à automatiser le transfert de pièces entre deux postes de stockage à l'aide de deux robots : un robot industriel FANUC Mi10 ou Mi20 et un robot didactique Niryo Ned2.

Installés dans un atelier bien équipé et sécurisé, ces robots sont coordonnés via un tapis roulant motorisé et une interface homme-machine (IHM) utilisant le protocole Modbus/TCP. Le robot FANUC prend des pièces du premier poste de stockage pour les placer sur le tapis roulant, qui les transporte jusqu'au second poste où le robot Niryo les récupère.

Supervisé par une IHM, ce système garantit une production efficace et en temps réel. Ce projet nous permet de mettre en pratique nos compétences en informatique et réseaux dans un environnement technique et collaboratif.

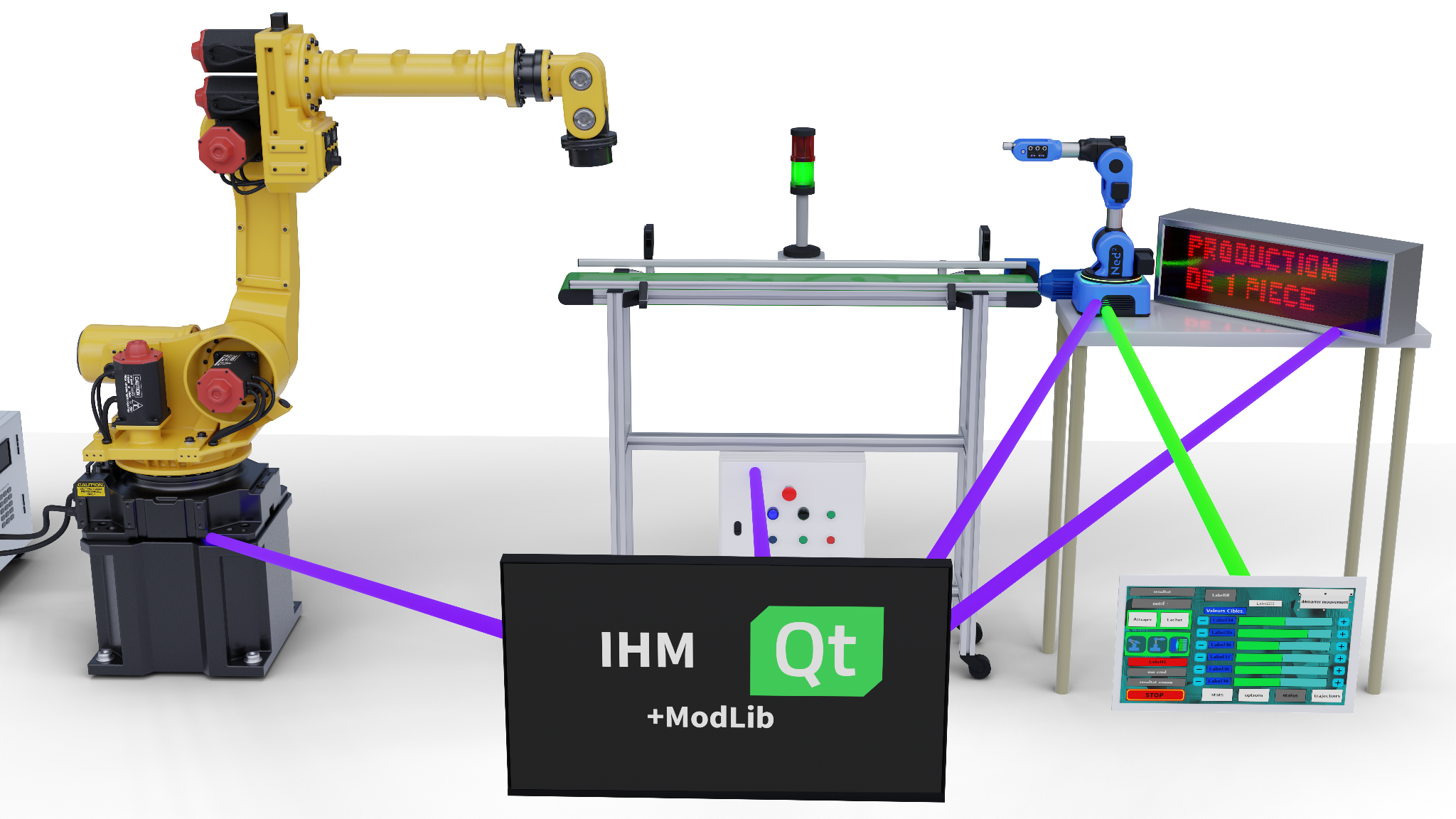
## 

## 1.2 Répartition des tâches

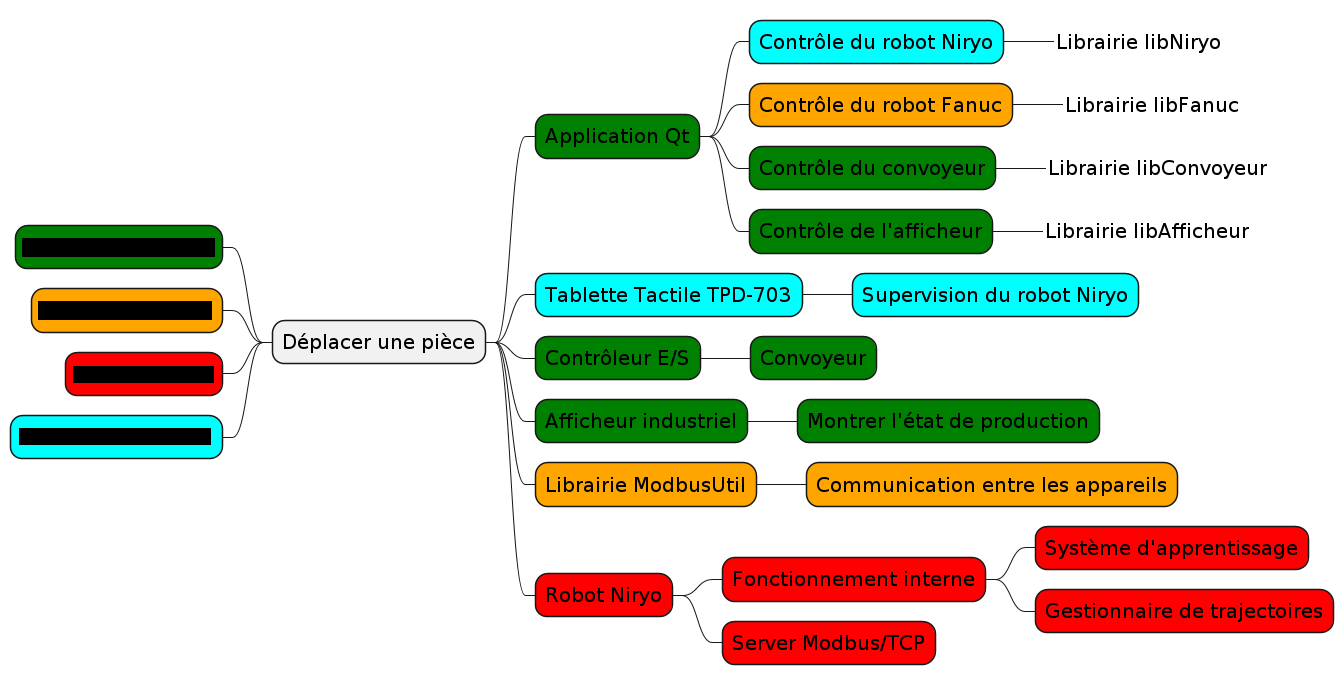
La réalisation de ce projet de chaîne de manutention robotisée requiert une forte collaboration entre les membres de l'équipe pour concevoir et développer divers éléments logiciels et matériels. Les tâches incluent la supervision globale de la partie opérative via une interface homme-machine (IHM), qui permet d'initialiser les matériels, de démarrer et d'arrêter le cycle de production, et de synchroniser les différents composants du système, comme le robot FANUC, le tapis roulant et le robot Niryo. Le contrôle et la commande du tapis roulant impliquent le développement des fonctions nécessaires pour gérer son mouvement, y compris les commandes de marche/arrêt, la direction du déplacement, et la gestion des capteurs de détection des pièces. Le contrôle du robot FANUC nécessite également le développement d'une bibliothèque Modbus/TCP en mode maître pour assurer une communication fluide entre les éléments du système. Pour le robot didactique Niryo Ned2, les tâches incluent son contrôle et sa commande, ainsi que la gestion de la base de données pour le stockage du plan mémoire Modbus/TCP. Enfin, le pilotage du robot Niryo via un écran tactile, en émulation de "Teach Pendant", et le développement de la bibliothèque Modbus/TCP en mode esclave permettent une communication efficace avec le système de supervision. Cette répartition assure que chaque aspect du projet est abordé de manière spécialisée et intégrée, permettant une réalisation efficace et coordonnée de la chaîne de manutention robotisée.

## 

## 



*Figure 1 : Rendue 3d du système à réaliser*



*figure 2: répartition des tâches*

## 1.3 cahier des charges personnelle

Mes tâches était donc les suivantes:

**F1 : IHM de supervision globale de la partie opérative**

• Application graphique

• Initialisation des matériels

• Départ cycle et mise en arrêt de la production

• Synchronisation des éléments de la PO : robot1, convoyeur, robot2

• Supervision des éléments de la PO : Affichage de leur état en numérique ou en synoptique

• Envoi des données de l'état de la production sur un afficheur industriel

**F2 : Contrôle - commande du convoyeur**

• Commande Marche/Arrêt, sens de déplacement, petite vitesse, grande vitesse

• Détection des pièces par deux capteurs CptPiece1 et CptPiece2 à chaque extrémité

• Pilotage des voyants de production Rouge/Vert

• Surveillance du variateur de vitesse avec convertisseur RS485/Ethernet

• Contrôle - commande par un boîtier Entrées/Sorties Modbus/TCP

**F9 : Affichage de l'état de la production**

• Afficheur industriel

• Affichage du mode de production en cours, données de production

• Communication avec le protocole Modbus/TCP

# 

## 1.4 Gestion de projet

Pour la gestion de notre projet, nous avons tous utilisé Trello, un outil de gestion de projet qui nous a grandement aidés dans notre organisation. Trello est extrêmement intuitif et facile à utiliser, ce qui nous a permis de s'y retrouver rapidement. Grâce à son interface visuelle basée sur des cartes et des listes, il a été simple de suivre l'avancement des tâches, de les assigner aux membres de l'équipe, et de voir en un coup d'œil ce qui était en cours et ce qui restait à faire.

La collaboration en temps réel a été un autre avantage majeur. Chacun pouvait ajouter des commentaires, des fichiers et des mises à jour sur les cartes, facilitant ainsi la communication et la coordination au sein de l'équipe. Trello a également prouvé sa flexibilité en s'adaptant à nos besoins spécifiques grâce à ses nombreuses options de personnalisation et d'intégration avec d'autres outils que nous utilisions.

Il permet aussi de générer des diagrammes de Gantt via des extensions appelées Power-Ups. Nous avons utilisé cette fonctionnalité pour créer notre diagramme de Gantt, qui se trouve en annexe 1. Ce diagramme nous a aidés à visualiser le planning global du projet et à gérer les dépendances entre les différentes tâches, assurant ainsi une gestion plus efficace et structurée de notre projet.

# 2. Situation dans le projet

## 2.1 Analyse des besoins

Ce diagramme (figure 3) permet de structurer les besoins en deux catégories distinctes : les besoins de l'utilisateur standard et ceux du technicien.

L’utilisation du système se divise en deux parties principales.

Premièrement, l’utilisateur standard doit pouvoir superviser et piloter la chaîne de manutention robotisée depuis l'interface homme-machine (IHM). Cette IHM offre des commandes visuelles intuitives et des informations en temps réel sur l'état des différents équipements du système, tels que le robot FANUC, le robot Niryo, et le tapis roulant. L'utilisateur peut ainsi initialiser les matériels, démarrer et arrêter le cycle de production, et surveiller la synchronisation des composants du système pour assurer un fonctionnement harmonieux et efficace.

Deuxièmement, le technicien, qui dispose d'un accès plus avancé, doit être capable de superviser et piloter le système de manière similaire à l'utilisateur standard, mais avec des capacités supplémentaires. En plus de l'utilisation de l'IHM pour la supervision standard, le technicien doit pouvoir contrôler directement l'infrastructure du système via une interface dédiée. Cette interface doit permettre de tester les fonctionnalités du convoyeur, gérer les capteurs de détection des pièces.

## 2.2 – Analyse des exigences

Voici une analyse détaillée des exigences de mon rôle.

### 2.2.1 Interface Homme-Machine (IHM) :

L'IHM doit être capable d'initialiser les matériels, vérifiant la disponibilité et l'état initial des composants tels que les robots, le tapis roulant et les capteurs. Elle doit permettre de démarrer et d'arrêter le cycle de production en synchronisant les différents éléments du système, comme le robot FANUC et le robot Niryo. L'IHM doit afficher en temps réel l'état des composants et du processus de production, permettant une surveillance continue et la détection rapide d'anomalies.

### 2.2.2 Contrôle du Tapis Roulant :

L'IHM doit gérer le mouvement du tapis roulant, incluant les commandes de marche/arrêt et la direction du déplacement. Elle doit également gérer les capteurs de détection des pièces positionnés sur le tapis roulant, assurant une détection fiable des pièces et signalant tout dysfonctionnement.

### 2.2.3 Afficheur Industriel :

L'IHM doit afficher l'état de la production sur un afficheur industriel, incluant des statuts de production clairs (en cours, en pause, arrêté) et des données quantitatives comme le nombre de pièces déplacées et le temps de production. Cela permet une évaluation précise de la performance et une visibilité immédiate sur l'état de la production.

### 2.2.4 Interface Technicien :

Une interface dédiée doit permettre aux techniciens de tester les fonctionnalités du convoyeur, incluant la marche/arrêt, la direction du déplacement et la gestion des capteurs. Cette interface est essentielle pour les opérations de maintenance et de diagnostic.

**Exigences Techniques :**

Le système doit utiliser le protocole Modbus/TCP pour assurer une communication fluide et sécurisée entre les différents composants. Des mesures de sécurité doivent être mises en place pour protéger les opérateurs et le matériel, notamment en cas d'arrêt d'urgence.

Cette analyse des exigences montre les fonctionnalités essentielles nécessaires pour la réussite du projet de chaîne de manutention robotisée. En tant qu'étudiant 1, ma responsabilité est de m'assurer que l'IHM, le contrôle du tapis roulant, l'afficheur répondent à ces exigences, garantissant ainsi un fonctionnement efficace et sécurisé du système.

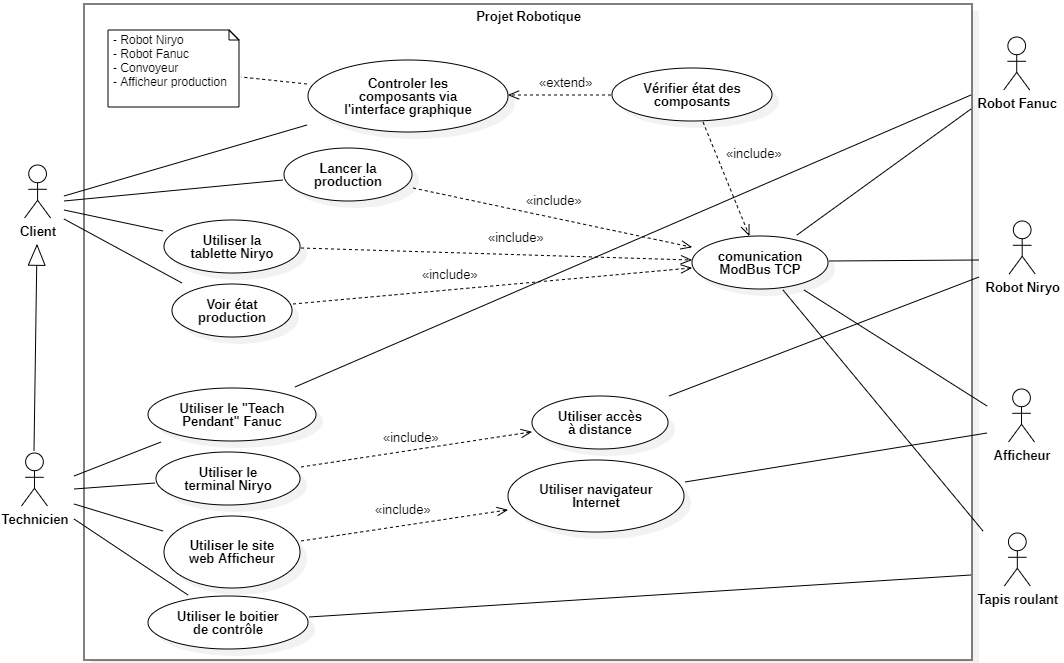


figure 3 : Cas d’utilisation du système

# 3. [Réalisation](#_6g8bqyc643r7)

## 3.1 C[onfiguration](#_v1hj26prdz8g)

Dans cette section, je vais parler des choix de configuration que j'ai faits pour la partie du projet dont j'étais responsable. Mon rôle était de superviser toute la partie opérative (PO), de contrôler le convoyeur, et de préparer l'affichage de l'état de la production.

Pour ce projet, les configurations étaient imposées par le sujet, donc je n'avais pas à choisir entre plusieurs options. J'ai suivi les instructions données. Même si je n'ai pas encore réalisé l'IHM de supervision globale ni l'afficheur industriel, j'ai mis en place une interface graphique pour un technicien. Cette interface permet de tester les fonctionnalités du convoyeur. J'ai utilisé la librairie Modbus de Qt pour contrôler les différentes fonctions du convoyeur.

Le convoyeur est équipé de capteurs CptPiece1 et CptPiece2 qui détectent la présence des pièces à chaque extrémité. Grâce à l'interface graphique, le technicien peut facilement commander la marche/arrêt du convoyeur, choisir le sens de déplacement, et régler la vitesse. La communication avec le convoyeur se fait via un boîtier Entrées/Sorties Modbus/TCP et un convertisseur RS485/Ethernet. Cela permet une intégration fluide avec le reste du système.

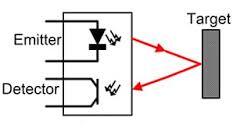
Même si l'IHM de supervision globale n'est pas encore terminée, l'interface graphique que j'ai développée pour le technicien est un premier pas important. Elle permet de tester et de valider les fonctionnalités du convoyeur de manière efficace. J'ai utilisé Qt 6 pour développer cette interface, car c'est un framework puissant et flexible qui s'intègre bien avec les autres composants du projet.

Concernant l'afficheur industriel, bien que je ne l'aie pas encore installé, il est prévu de l'utiliser pour montrer l'état de la production en temps réel. L'afficheur industriel MAT AIML sera connecté au système de supervision via le protocole Modbus/TCP. Il affichera le mode de production en cours et les données de production. Une fois mis en place, il permettra une mise à jour constante et précise des informations affichées, ce qui est crucial pour assurer une surveillance efficace de la chaîne de manutention.

Le protocole Modbus/TCP était imposé par le sujet parce qu'il est robuste et compatible avec les composants matériels utilisés. Il permet une communication efficace entre le serveur de supervision et les robots, ce qui assure une bonne coordination des opérations. Les configurations spécifiques incluent la gestion des signaux UOP pour le robot FANUC et la communication en mode esclave pour le robot Niryo Ned2.

## 3.2 Matériels mis à disposition

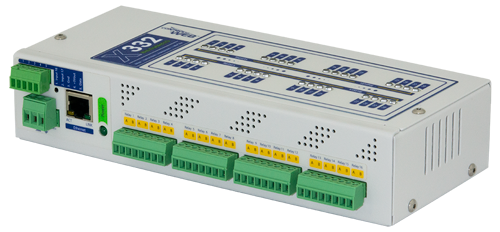
J'avais à ma disposition plusieurs équipements physiques essentiels pour réaliser mes tâches de supervision de la partie opérative (PO), de contrôle-commande du convoyeur, et de préparation de l'affichage de l'état de la production. Le convoyeur, élément central de la chaîne de manutention, est conçu pour transporter des pièces d'un point A à un point B. Il est équipé de capteurs de proximité infrarouge Crouzet placés aux extrémités du convoyeur pour détecter la présence des pièces. Ces capteurs infrarouges fonctionnent en émettant un faisceau qui, lorsqu'il est interrompu par une pièce, envoie un signal au système de contrôle (TOR).



*figure 4 : fonctionnement capteurs de proximité*

Pour contrôler le convoyeur, j'ai utilisé un boîtier Entrées/Sorties Modbus/TCP, modèle X-332-24I , qui gère les entrées et sorties numériques nécessaires pour commander les différentes fonctions du convoyeur. Ce boîtier dispose de 16 entrées numériques isolées et de 16 sorties relais, permettant de commander la marche/arrêt du convoyeur, de définir le sens de déplacement et de régler la vitesse. Chaque entrée et sortie est isolée pour protéger contre les interférences électriques, assurant ainsi la fiabilité des signaux.

La communication entre le boîtier Entrées/Sorties et le reste du système se fait via Ethernet.



*figure 5 : Contrôleur x-332*

Bien que l'afficheur industriel ne soit pas encore installé, il est prévu de l'utiliser pour afficher l'état de la production en temps réel. L'afficheur industriel MAT AIML est conçu pour les environnements industriels, il offre une visibilité claire et immédiate des données de production. Il possède un écran LED haute luminosité, visible même dans des conditions de faible éclairage . Il permet d’afficher sur 2 lignes de 12 caractères un message de 40 caractères en mode fixe ou clignotant (seulement 24 caractères seront visibles) ou encore en mode défilant (les 40 caractères seront visibles) .



*figure 6 : afficheur industrielle MAT AIML*

## 3.3 Câblage Boîtier E/S

Quand j'ai réalisé le câblage de ce système, j'ai commencé par alimenter le module X-332 en 24V. Cette alimentation est essentielle pour fournir une énergie suffisante à tous les capteurs et relais connectés. Ensuite, j'ai connecté le module au réseau local via un câble RJ45, en utilisant l'adresse IP 192.168.1.122:502 pour s’y connecter le port 502 correspond au port de communication ModBus. Cette connexion permet la communication et le contrôle de l'automate.

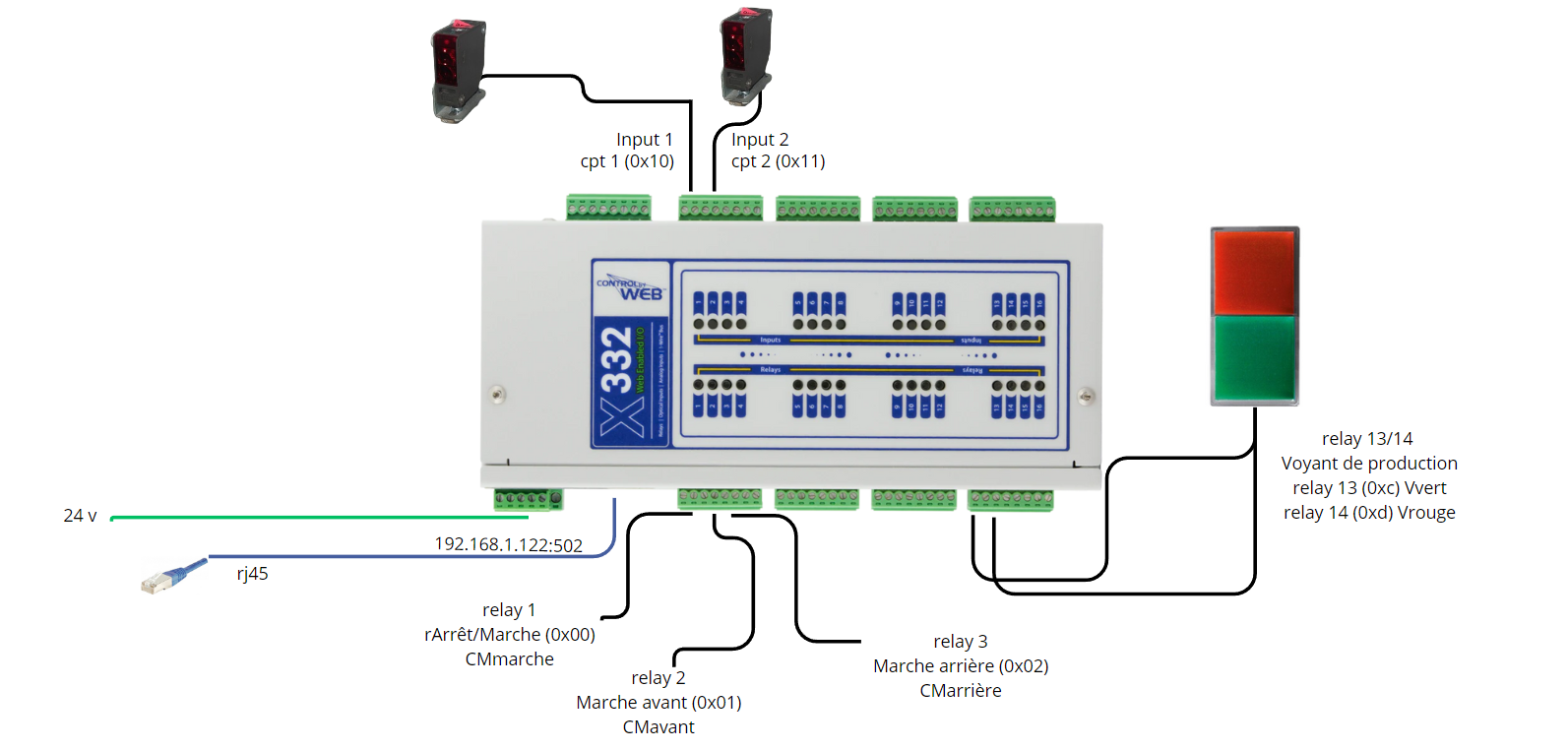
Pour les entrées de capteurs, j'ai relié deux capteurs aux entrées Input 1 et Input 2 du module, aux adresses 0x10 et 0x11 respectivement. Ces capteurs sont cruciaux pour déclencher des actions automatisées en fonction des événements qu'ils détectent.

Ensuite, j'ai câblé les relais pour les commandes du moteur. Le relais 1, à l'adresse 0x00, contrôle l'état général de marche ou d'arrêt du système. Il permet d'activer ou de désactiver le système de manière centrale. Le relais 2, à l'adresse 0x01, est utilisé pour contrôler le sens de marche avant du moteur, tandis que le relais 3, à l'adresse 0x02, gère le sens de marche arrière. Ces relais permettent de commander la rotation du moteur dans les deux directions.

Enfin, j'ai câblé les relais 13 et 14 pour le voyant de production, avec le relais 13 (adresse 0x0c) pour le voyant vert et le relais 14 (adresse 0x0d) pour le voyant rouge. Ces voyants fournissent une indication visuelle de l'état de production, comme un état opérationnel ou une alerte.

En connectant soigneusement chaque composant (figure 7) , j'ai assuré un contrôle précis et fiable du système. L'automate peut ainsi recevoir des requêtes ModBus, traiter ces informations et déclencher des actions appropriées via les relais. Cette configuration est typiquement utilisée dans des environnements industriels pour automatiser et surveiller des

processus de production.



*figure 7 : Câblage boîtier E/S présent dans le convoyeur*

## 3.4 logiciels et langages de programmation



J'ai utilisé Qt/c++ pour développer les interfaces de commande et de supervision de notre système automatisé. L'un des aspects cruciaux de ce projet était l'utilisation du protocole Modbus/TCP pour la communication entre les différents composants du système, comme les robots et le convoyeur.

Le protocole Modbus/TCP est un protocole de communication standard utilisé dans l'industrie pour permettre la transmission de données sur des réseaux TCP/IP. Il est particulièrement adapté aux environnements industriels en raison de sa simplicité et de sa fiabilité. Dans notre projet, il permet de superviser et de contrôler les différents éléments du système via le réseau local.

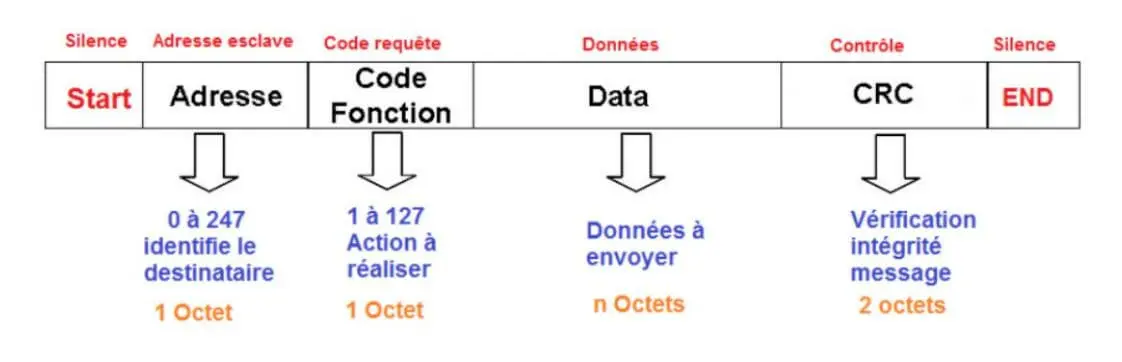
Pour intégrer Modbus/TCP dans notre application Qt, j'ai utilisé des bibliothèques spécifiques disponibles pour Qt. Ces bibliothèques fournissent les fonctionnalités nécessaires pour établir des connexions réseau, envoyer des requêtes et recevoir des réponses ce qui représente un gain de temps non négligable. Voici comment j'ai procédé :

**Établissement de la connexion :**

J'ai utilisé les classes `QTcpSocket` et `QModbusTcpClient` de Qt pour créer une connexion TCP/IP avec les dispositifs ciblés. La classe `QTcpSocket` permet de gérer les connexions réseau de manière simple, tandis que `QModbusTcpClient` est spécialement conçue pour le protocole Modbus/TCP.

**Envoi de requêtes Modbus :**

Les requêtes Modbus sont des messages structurés selon le protocole. Par exemple, pour lire des registres ou écrire des valeurs, j'ai utilisé des fonctions comme `readHoldingRegisters` ou `writeSingleRegister` fournies par `QModbusTcpClient`. Ces fonctions génèrent automatiquement les requêtes conformes au protocole Modbus/TCP et les envoient aux dispositifs.



*figure 8 : exemple requête ModBus*

**Réception et traitement des réponses :**

Les réponses des dispositifs sont reçues via les mêmes classes. Qt gère les réponses de manière asynchrone, ce qui signifie que notre application n'est pas bloquée en attendant les réponses. J'ai utilisé des signaux et des slots pour traiter les réponses dès qu'elles sont reçues. Par exemple, lorsqu'une réponse est reçue, un signal est émis et connecté à un slot qui traite les données reçues.

**Afficheur industrielle:**

N’aillant pas commencé la partie afficheur, J’ai utilisé le logiciel de programmation AIMLP511 qui est conçu pour tester et configurer l'afficheur industriel AIMLP50 . Il permet de créer, modifier et transférer des messages via une connexion série ou Ethernet. Ce logiciel, compatible avec divers systèmes Windows, offre une interface intuitive pour gérer jusqu'à 510 messages, permettant de les afficher en mode fixe, clignotant ou défilant. Bien que l'intégration finale avec Qt soit prévue ultérieurement, ce logiciel est essentiel pour vérifier le bon fonctionnement des afficheurs avant leur utilisation définitive.

En résumé, l'utilisation de Qt et du protocole Modbus/TCP a permis de créer des interfaces graphiques interactives et fiables pour superviser et contrôler notre système automatisé. Les bibliothèques Qt facilitent grandement la gestion des connexions réseau et des échanges de données, rendant le développement plus efficace et rapide.

## 3.5 Contrôle-Commande du Convoyeur

### 3.5.1 Communication

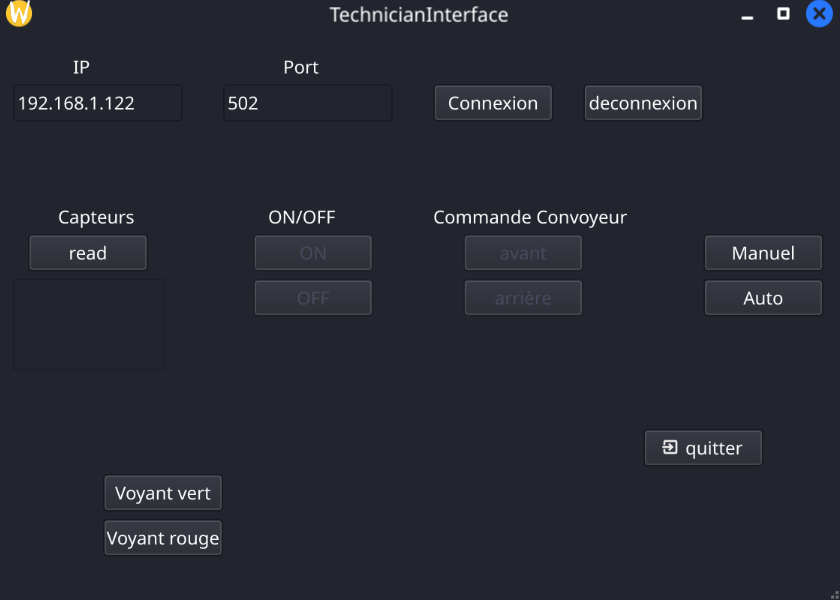
Dans ce projet, deux solutions de conception pour communiquer avec le convoyeur ont été considérées : créer une bibliothèque de communication personnalisée ou utiliser une bibliothèque Modbus existante (QModbus). La première offre un contrôle total et une personnalisation complète mais nécessite beaucoup de temps et de travail. En revanche, utiliser la bibliothèque Modbus est plus rapide et fiable, car elle est déjà bien documentée et soutenue. Bien que moins flexible, cette solution est préférable car elle simplifie le développement et assure une robustesse immédiate.

### 3.5.2 Fonctionnalité

J'ai développé une interface utilisateur dédiée aux techniciens pour tester et contrôler les fonctionnalités du convoyeur que j'ai mises en place. Cette interface permet au technicien de se connecter à un automate X-332 via son adresse IP et le port Modbus, établissant ainsi une communication pour interagir avec le système.

Une fois connecté, le technicien peut choisir entre deux modes de fonctionnement : le mode manuel et le mode automatique. En mode manuel, le technicien peut tester directement les différentes commandes disponibles pour le convoyeur. Il peut démarrer ou arrêter le convoyeur, le faire avancer ou reculer, et vérifier si des pièces sont détectées grâce aux capteurs intégrés. Les voyants verts et rouges permettent au technicien de s'assurer que le convoyeur et ses fonctionnalités fonctionnent correctement.

En mode automatique, le technicien peut lancer un algorithme prédéfini pour gérer le fonctionnement du convoyeur. Cet algorithme prend en charge les différentes tâches et assure une gestion optimale du système.



*figure 9 : interface technicien Convoyeu*r

### 3.5.3 Mode automatique

Dans l'interface technique que j'ai développée, le mode automatique gère le fonctionnement du convoyeur en utilisant deux capteurs (capteur1\_etat et capteur2\_etat) et une variable de contrôle (productionTermine). Voici comment il fonctionne :

**Déplacement du convoyeur en marche avant** :

Si le capteur 1 détecte une pièce (capteur 1 est activé) et que le capteur 2 ne détecte pas de pièce (capteur 2 est désactivé), le convoyeur avance. La production n'est pas encore terminée (productionTermine est faux).

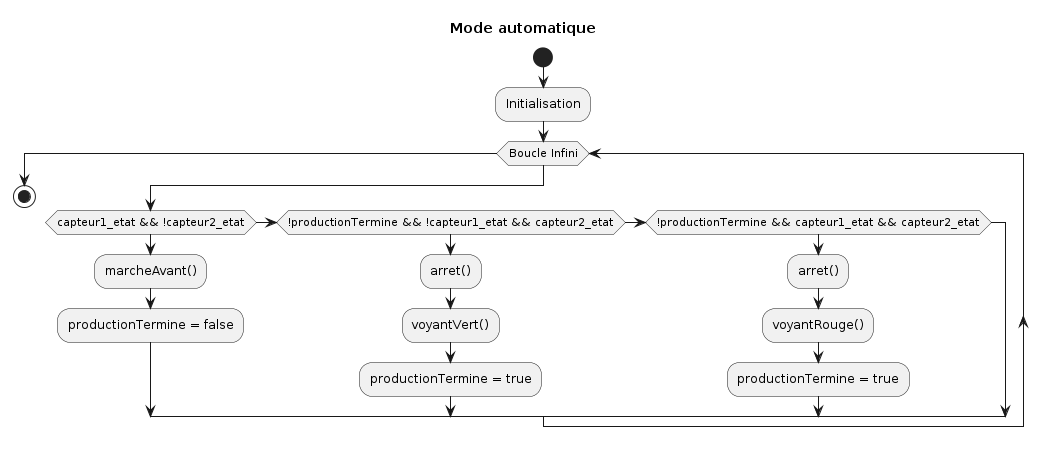
**Arrêt du convoyeur et voyant vert** :

Si la production n'est pas terminée et qu'une pièce n'est plus détectée par le capteur 1 mais est détectée par le capteur 2, le convoyeur s'arrête et le voyant vert s'allume, indiquant que la production est terminée avec succès. La variable productionTermine est mise à vrai.

**Arrêt du convoyeur et voyant rouge** :

Si la production n'est pas terminée et que les deux capteurs détectent simultanément une pièce, cela signale une erreur. Le convoyeur s'arrête et le voyant rouge s'allume, indiquant une erreur. La variable productionTermine est mise à vrai.

En résumé, le mode automatique contrôle le convoyeur pour avancer, s'arrêter avec succès, ou s'arrêter en cas d'erreur, en fonction des signaux des capteurs et de l'état de la production.



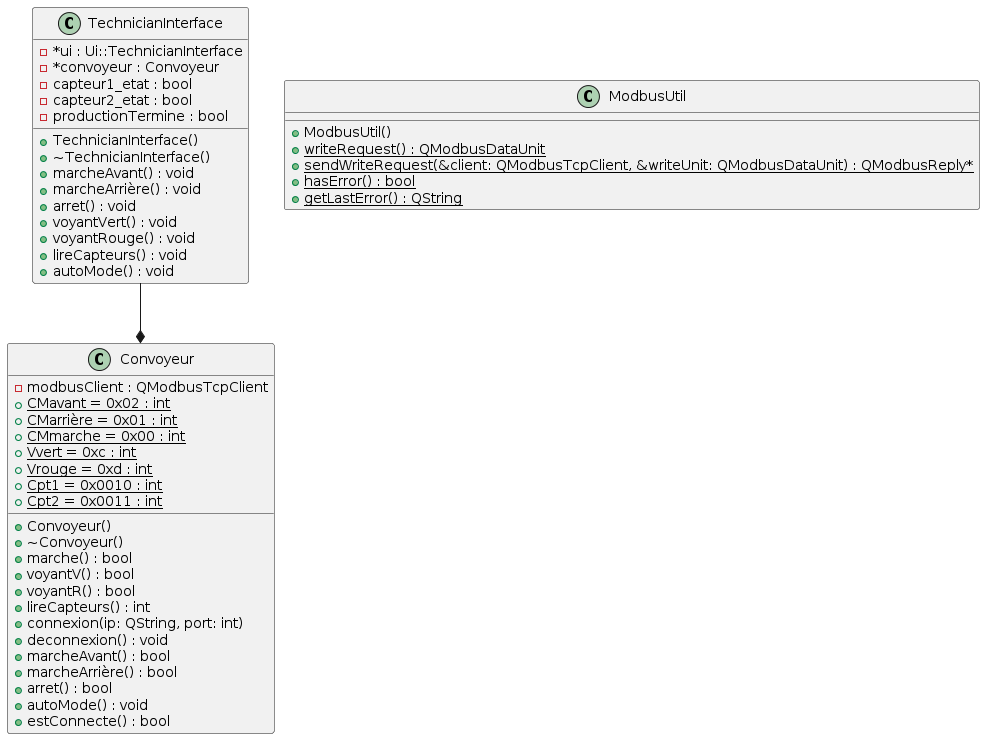
*figure 10 : diagramme d’activité Mode Automatique*

### 3.5.4 Partie Programmation

**Arborescence Fichiers Convoyeur**

| Convoyeur  ├── convoyeur.cpp  ├── convoyeur.h  ├── Convoyeur.pro  ├── Convoyeur.pro.user  ├── main.cpp  ├── modbusutil.cpp  ├── modbusutil.h  ├── technicianinterface.cpp  ├── technicianinterface.h  └── technicianinterface.ui |
| --- |

**Diagramme de classe**



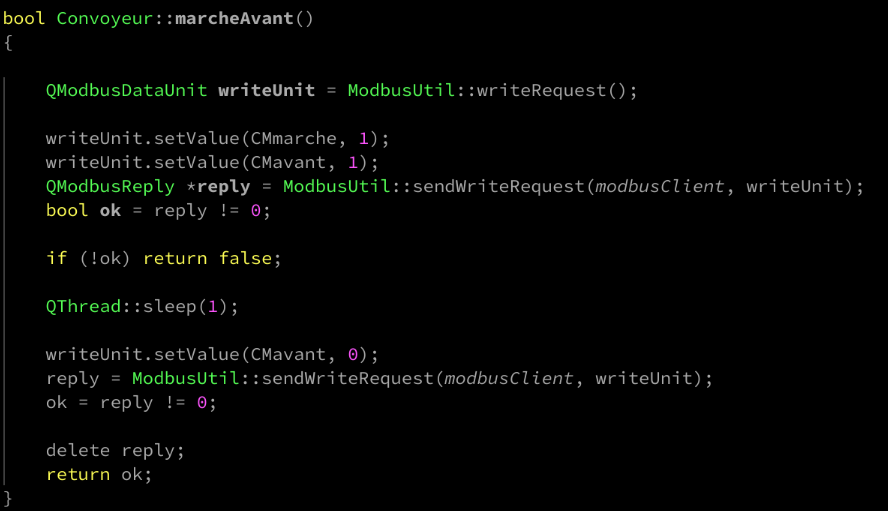
*figure 11 : diagramme de classe*

Ce diagramme a été réalisé avec PlantText . C’est un outil en ligne qui permet de générer rapidement des images à partir d'un texte. Il est principalement utilisé pour générer des images UML . Le prompt réalisé pour cette génération est en annexe .

**Convoyeur**

La classe `Convoyeur` est utilisée pour contrôler un convoyeur avec le protocole Modbus. Elle se connecte au serveur Modbus, envoie des commandes pour démarrer ou arrêter le convoyeur, et lit les données des capteurs. Des fonctions pour faire avancer, reculer, arrêter le convoyeur, et le mettre en mode automatique. Elle gère également la connexion et la déconnexion du serveur.

**Exemple :**



*figure 12 : Fonction marche avant*

Cette fonction permet de faire avancer un convoyeur en utilisant le protocole Modbus pour envoyer des commandes à un dispositif.

**Initialisation de la commande Modbus :**

QModbusDataUnit writeUnit = ModbusUtil::writeRequest();

”writeUnit` est une unité de données Modbus initialisée pour écrire des commandes.

**Définition des valeurs à envoyer :**

writeUnit.setValue(CMmarche, 1);

writeUnit.setValue(CMavant, 1);

- `CMmarche` et `CMavant` sont des constantes ou des définitions représentant des registres Modbus spécifiques. Ici, ils sont mis à `1` pour activer la marche avant.

**Envoi de la commande Modbus :**

QModbusReply \*reply = ModbusUtil::sendWriteRequest(modbusClient, writeUnit);

- La commande est envoyée au client Modbus (`modbusClient`).

- `reply` est un pointeur vers la réponse de la requête.

**Vérification de la réponse :**

bool ok = reply != 0;

if (!ok) return false;

- Vérifie si la réponse est valide. Si ce n'est pas le cas, la fonction retourne `false`.

**Pause pour permettre l'exécution :**

QThread::sleep(1);

- l’Automate x-332 effectue un contact, l’adresse du dispositif n’est donc pas obligé de rester à 1 indéfiniment pour fonctionner ici le fonction attends 1 seconde avant de continuer .

writeUnit.setValue(CMavant, 0);

reply = ModbusUtil::sendWriteRequest(modbusClient, writeUnit);

ok = reply != 0;

- `CMavant` est remis à `0` pour arrêter le contact mais la marche avant est toujours active.

- La commande est envoyée à nouveau et la réponse est vérifiée.

**Nettoyage et retour du résultat :**

delete reply;

return ok;

- Libère la mémoire utilisée par `reply`.

- Retourne `true` si tout s'est bien passé, sinon `false`.

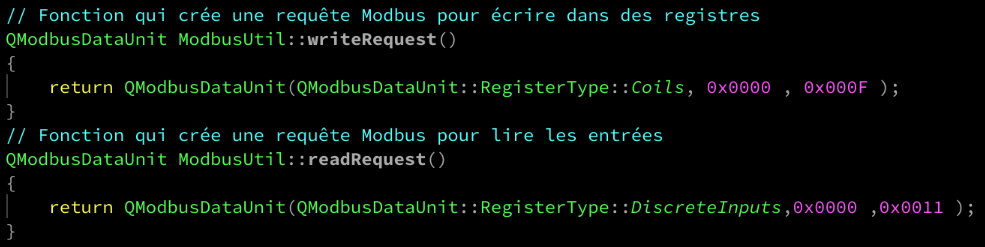
**TechnicianInterface**

La classe `TechnicianInterface` est une interface utilisateur pour contrôler le convoyeur. Elle affiche l'état des capteurs, permet de démarrer et arrêter le convoyeur et de passer en mode automatique. Elle inclut une instance de la classe `Convoyeur` pour exécuter ces actions. L'interface utilisateur comporte des boutons et autres éléments interactifs pour permettre au technicien de contrôler facilement le convoyeur.

**ModbusUtil**

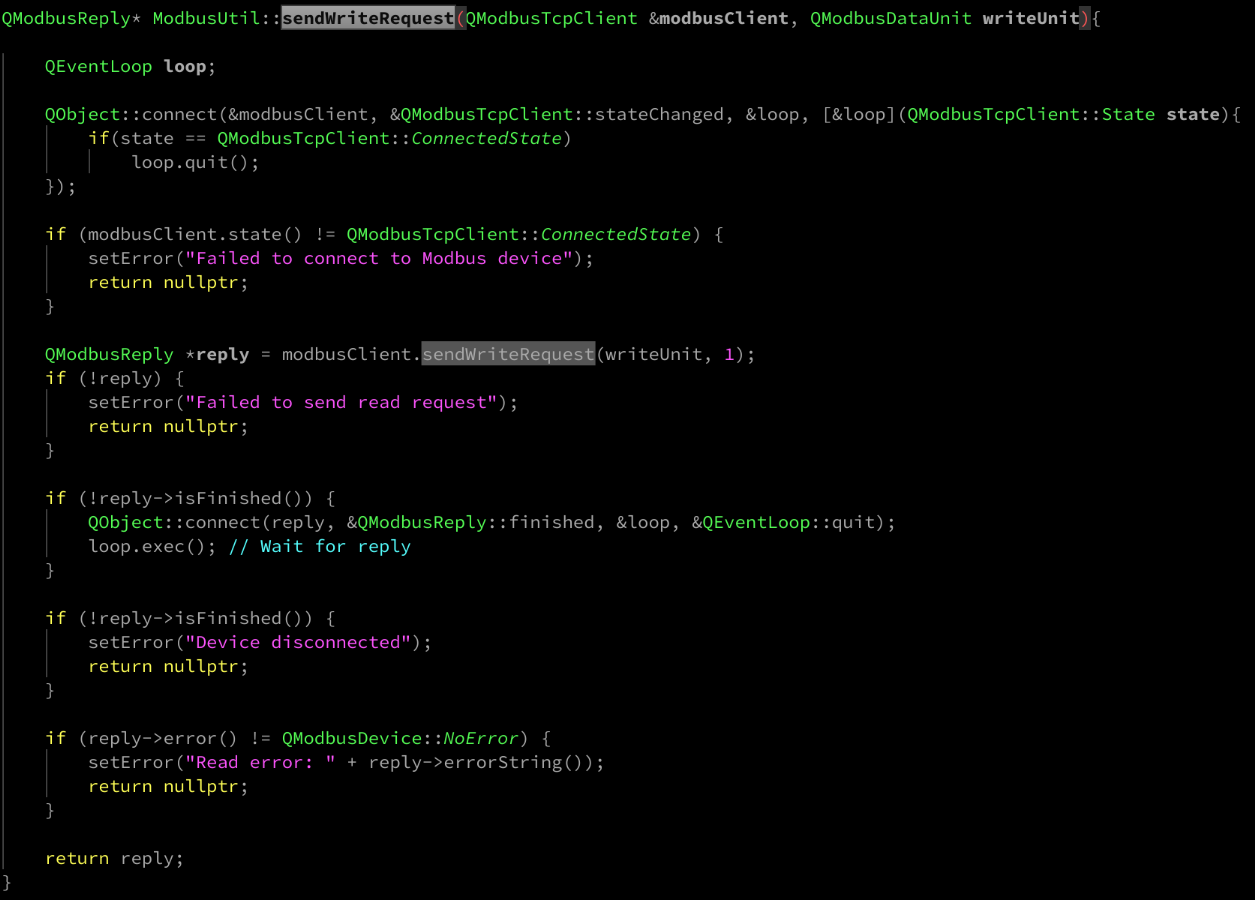
La classe ModbusUtil simplifie l'interaction avec les dispositifs Modbus dans une application Qt. Elle permet de créer des requêtes d'écriture, d'envoyer des requêtes via un client Modbus et de gérer les erreurs de communication.

**exemple :**

****

*figure 13 : fonction création requête*

C’est fonctions sont essentiel à la création de nos requête Modbus d'écriture et de lecture, une fois la requête créée il faut l’envoyer, pour cela deux fonctions sont utilisé, L’envoie d’une requête d’écriture “**sendWriteRequest**” et l’envoie d’une requête de lecture “**sendReadRequest**”



*figure 14 : Envoi de requête d’écriture*

**Module utilisé**

Afin de pouvoir profiter de certaines classe il est nécessaire de rajouter certains module dans le fichier .pro du projet voici le mien :

QT += core gui network serialbus

Le module serial bus permet de travailler avec des bus série tels que CAN et Modbus . Network fournit des classes pour les communications réseau . gui est un module nécessaire pour toutes les applications qui ont une interface graphique . Chaque module présent est essentiel au bon fonctionnement du programme .

## 3.6 Afficheur industrielle

Cette partie n’étant pas commencé les informations suivantes ne sont que théorique et ne relève pas d’une tâche déjà réalisé

L'afficheur industriel est essentiel pour montrer en temps réel les informations sur la production. Il indique des données importantes de la production. Cela permet aux techniciens de surveiller facilement le système et de réagir rapidement si besoin.

Pour commencer je vais le configurer pour qu'il puisse recevoir et afficher les données . Cela veut dire définir quels registres Modbus contiendront les informations à afficher, comme le statut de production,les états de marche/arrêt du convoyeur.

Je vais créer une interface graphique pour l'afficheur industriel en utilisant QT. Elle doit être simple et montrer clairement les données de production. Elle doit aussi être réactive et mettre à jour les informations en temps réel pour que les techniciens aient toujours les dernières données sous les yeux.

Après avoir développé cette interface, je vais intégrer l'afficheur industriel dans le système global et tester la communication avec les autres composants. Il est très important de faire ces tests pour s'assurer que l'afficheur fonctionne bien et que les informations sont correctes et à jour.

Mon boulot c'est de faire en sorte que l'afficheur industriel affiche bien les données de production en temps réel, ce qui aide les techniciens à surveiller et à maintenir le système de manutention robotisée.

## 3.7 IHM Globale

Ma mission est de créer l'IHM (Interface Homme-Machine) globale. Cette interface est importante parce qu'elle permet de superviser et de contrôler tout le système, qui inclut un convoyeur et deux robots : un robot industriel FANUC et un robot didactique Niryo Ned2. Comme la totalité du projet L'IHM utilise le protocole Modbus/TCP pour tout gérer.

Pour commencer, je vais utiliser un outil de développement comme Qt 6, comme on l'a mentionné dans notre projet. Mon objectif est de concevoir une interface utilisateur qui soit à la fois intuitive et réactive. L'IHM doit afficher clairement l'état des robots, du convoyeur, et des capteurs. Elle doit aussi permettre aux techniciens de démarrer et d'arrêter la production.

Ensuite, il faut que je configure l'IHM pour qu'elle puisse bien communiquer avec les différentes parties du système en utilisant Modbus/TCP. Ça veut dire importer les bibliothèques de chaque partie du projet (Convoyeur, les deux robots et l’afficheur).

Après avoir développé l'interface utilisateur, je vais l'intégrer dans le système global. Il est super important de tester cette intégration pour s'assurer que tout fonctionne bien et que les infos affichées sont correctes et à jour. Ces tests vérifieront que l'IHM peut envoyer et recevoir les bonnes commandes et données, et qu'elle réagit de manière appropriée aux différents états du système.

## 3.8 Tests unitaires

Afin de valider chaque partie de mes tâches des test sont réalisés pour cela j’ai utilisé des tableaux de test unitaire que vous pouvez retrouver en annexes. Les parties F1 et F9 n’étant toujours pas abouties, seulement les tests unitaires de la partie F2 ont été réalisés. Les tests ont tous abouti à une validation du résultat attendu .

# 4. Bilan des travaux

**IHM de supervision globale :**

* Objectifs non atteints :
  + Je n'ai pas commencé le développement de l'IHM globale pour la supervision du système.

**Contrôle-commande du convoyeur :**

* Objectifs atteints :
  + J'ai développé une interface graphique pour le contrôle manuel et automatique du convoyeur.
  + J'ai mis en place la gestion des capteurs pour la détection des pièces.
  + Les fonctions de marche avant, marche arrière, et arrêt du convoyeur ont été implémentées et testées.
  + J'ai utilisé un boîtier E/S Modbus/TCP pour la communication avec le convoyeur.
* Travaux restants :
  + Intégration complète avec l'IHM globale.

**Afficheur industriel :**

* Objectifs non atteints :
  + Je n'ai pas encore installé ni configuré l'afficheur industriel.
  + Le développement de l'interface graphique pour l'afficheur n'a pas été commencé.
  + L'intégration avec le système de supervision pour afficher les données de production en temps réel reste à faire

**Analyse des Difficultés:**

**Problèmes techniques :**

* + La complexité de la communication Modbus/TCP a nécessité une compréhension approfondie des protocoles réseau.
  + La gestion des capteurs et des relais pour le convoyeur a présenté des défis en termes de synchronisation et de traitement en temps réel.

**Compétences Développées**

1. **Techniques:**

* Maîtrise de Qt/C++ pour le développement d'interfaces graphiques.
* Compréhension du protocole Modbus/TCP et de sa mise en œuvre.
* Expérience dans le câblage et la configuration de boîtiers E/S et de capteurs.

1. **Résolution de problèmes:**

* Capacité à identifier et résoudre des problèmes techniques.
* Adaptation des solutions en fonction des contraintes du projet.

**Conclusion**

Ce projet m'a permis de développer des compétences en informatique dans un environnement technique. Bien que certains objectifs restent à atteindre, notamment le développement de l'IHM globale et l'afficheur industriel, les progrès réalisés sont significatifs. Les tâches restantes seront abordées dans les jours suivants .

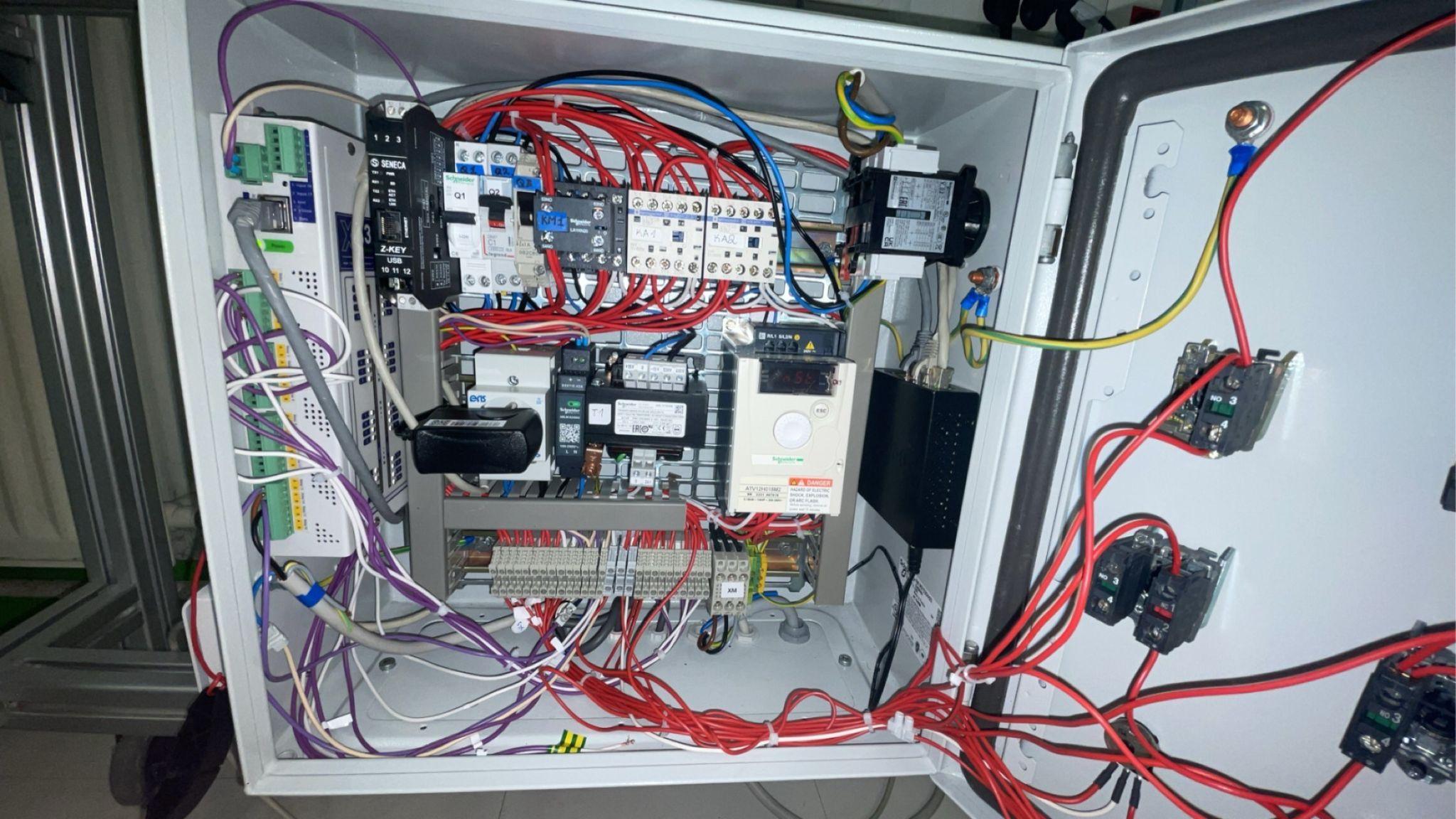
# 5. Annexes



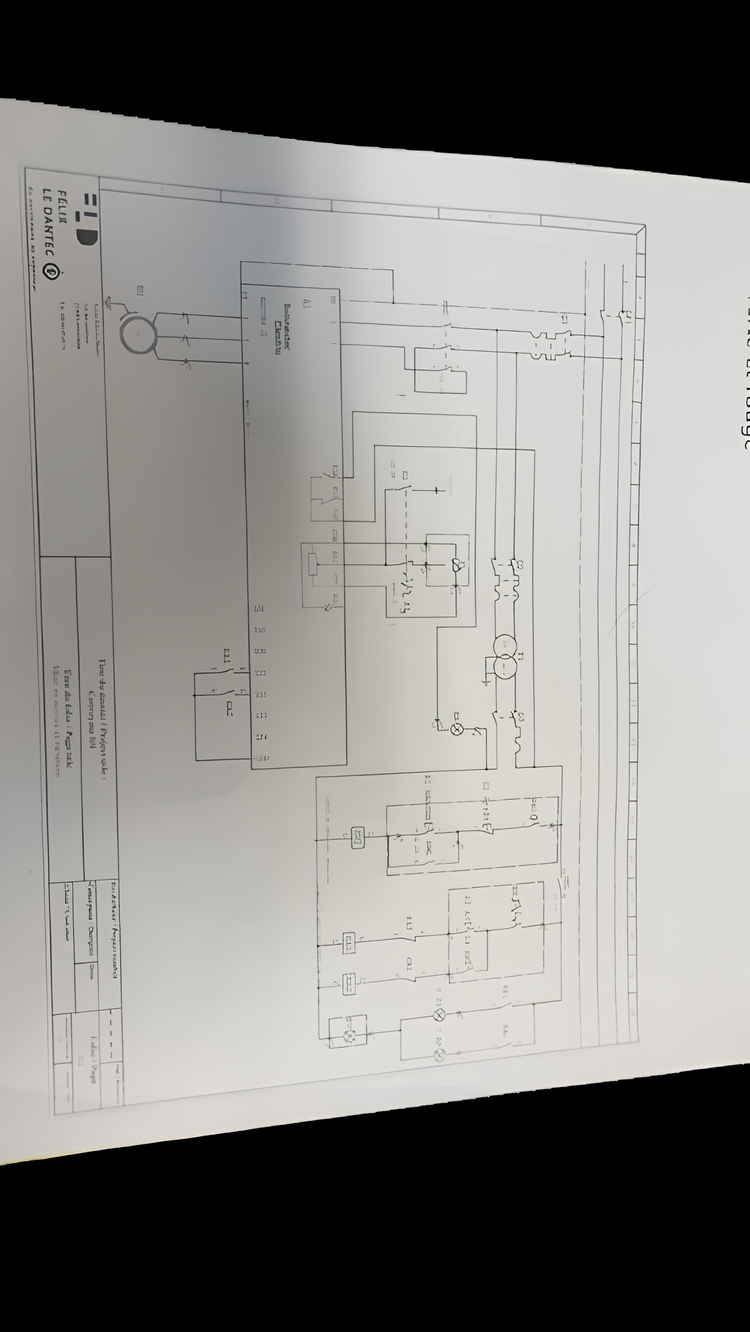
*annexe 1 : diagramme de gantt généré avec trello*



*annexe 2 : Convoyeur*



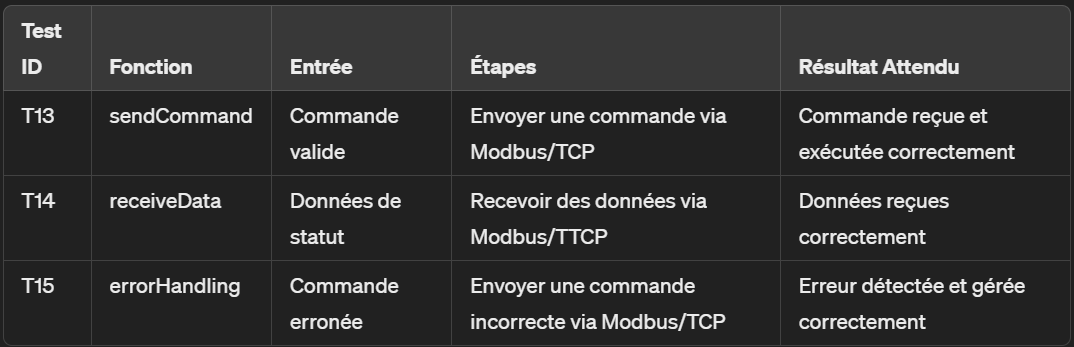
*annexe 3 : boitier électrique du Convoyeur*



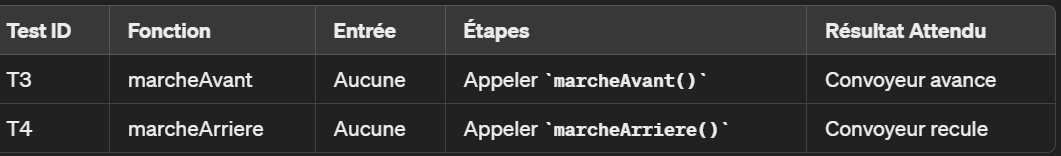
*annexe 4 : schéma electrique Convoyeur*



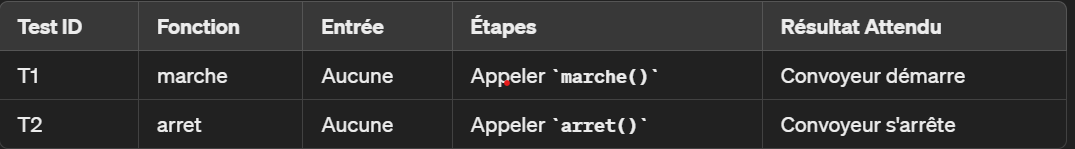
*Annexe 5 : test unitaire Capteurs ✅*



*annexe 6 : test unitaire Modbus/TCP ✅*



*annexe 7 : test unitaire Sens de déplacement✅*



*annexe 8 : test unitaire Marche/Arrêt✅*