

**Rapport de projet :**

**Projet Robotique**



**Partie Personnel**

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

**BTS SNIR**

**Lycée Félix Le Dantec**

[**Introduction 3**](#_p8apadpw047w)

[Explication du projet 3](#_pwa4ejg2ophw)

[**1 - Objectif de réalisation dans ce projet 4**](#_k3abesheij9d)

[Figure 1 : Vue 3D du Projet 4](#_t69oou9bzwf)

[**2 - Présentation des tâches à réaliser 5**](#_lieckfyf71zw)

[Figure 2 : Ensemble du projet 5](#_544u5v55phpb)

[2.1 - Tâches personnels 6](#_abr1w68e7at7)

[**3 - Situation dans le projet 7**](#_pknolfay15fu)

[Figure 3 : Diagramme de Cas d’Utilisation 7](#_6ul9fohif5rd)

[3.1 - Analyse du cahier des charges 8](#_oynuqdc8zct9)

[Figure 4 : Diagramme de déploiement 9](#_c3a5ogukuz7b)

[3.2 - Matériels et Logiciels 10](#_1jk0qykdbzva)

[3.2.1 - QT Creator 10](#_l8nsf3fov1tp)

[Figure 5 : IDE QT Creator 11](#_pgcebg2irfqz)

[3.2.2 - Télécommande “touch pendant” 12](#_s2qkz2kl7erk)

[Figure 6 : Télécommande Fanuc Mi10 13](#_ul25jjtgap8t)

[3.2.3 - Robot industriel FANUC MI10 13](#_m9j2al98u5ny)

[Figure 8 : Robot Fanuc MI10 15](#_b7cvub2scvdz)

[3.4 - Planification des travaux 15](#_sicztsvo4qmd)

[Figure 7 : Diagramme de Gant 16](#_v3833rk0q6wf)

[**4 - Conception du programme du Fanuc 17**](#_m6pr80q6nkyo)

[4.1 - Langage utilisé 17](#_545caa5csrrx)

[4.1.1 - Avantage du c++ 17](#_f8qio586yxdo)

[4.1.2 - Requête modbus 18](#_37m6d55mwvg5)

[4.2 - Arbre du projet Fanuc 19](#_1z7wyycqpm3p)

[Figure 8 : Arbre du projet Fanuc 19](#_e6ejmt3f7bir)

[Figure 9 : IHM graphique de test du projet FANUC 20](#_js537le5shkd)

[4.2.1 - UOP 20](#_r5e2u738ik5l)

[4.3 Problèmes rencontrés 21](#_bonhnvmlgf6)

[4.4 Diagramm de Classe du robot FANUC 22](#_1uu9pov4fttg)

[Figure 10 : Diagramme de classe FANUC 22](#_wx0qq6xn1czx)

[**5 - Conception du programme de modlib 23**](#_sqc31asb941a)

[5.1 - Utilité et Avantages de modlib 23](#_joe55e6554h3)

[5.2 - Arbre du projet modlib 24](#_aqlxq18yfg08)

[Figure 11 : Arbre du projet modlib 24](#_wy7ub72nfd3m)

[5.3 - Problèmes rencontrés 25](#_891co2fpfo01)

[5.4 - Diagramme de classe du projet modlib 26](#_hvklv1gsawjv)

[Figure 12 : Diagramme de classe modlib 26](#_jauamkmzj6va)

[**6 - Tests Unitaires FANUC 27**](#_1p8nepqervnr)

[**7 - Test Unitaire modlib 28**](#_s6q8ww79qk8u)

[**Synthèse des tâches réalisées 30**](#_764q0m2x115f)

[Bilan technique 30](#_vuhig84n9kbt)

[Bilan Personnel 30](#_jm9ok98hlkk8)

[Perspective du projet 30](#_ohtyu2q9zgqx)

[**Sitographie 31**](#_nc8acm334yjs)

[**Annexe 32**](#_2xrh0a2k6v9f)

[Code du Fanuc : 32](#_kp01ds62hnmx)

[Code du modlib : 49](#_qsskjic7gi9x)

# 

# 

# 

# 

# 

# 

# 

# 

# Introduction

## Explication du projet

Le projet vise à automatiser le processus de manutention de pièces entre deux postes de stockage, appelés "Poste de stockage 1" et "Poste de stockage 2". L'objectif est de rendre ce processus plus efficace, précis et sécurisé.

Le système se compose principalement d'un convoyeur et de deux robots. Le convoyeur joue un rôle crucial en transportant les pièces d'un poste de stockage à l'autre. Les robots, quant à eux, sont responsables de la manipulation des pièces. Il y a deux types de robots dans ce système : le robot industriel FANUC Mi10 (ou Mi20) et le robot didactique[[1]](#footnote-0) Niryo Ned2.

Le robot FANUC, déjà installé dans l'atelier du lycée, est conçu pour effectuer des tâches de manutention complexes et lourdes avec une grande précision. Le Niryo Ned2 est un robot plus petit, qui sera contrôlé via un écran tactile utilisant le protocole de communication Modbus/TCP. Ce protocole permet une communication fiable et standardisée entre le robot et le système de contrôle.

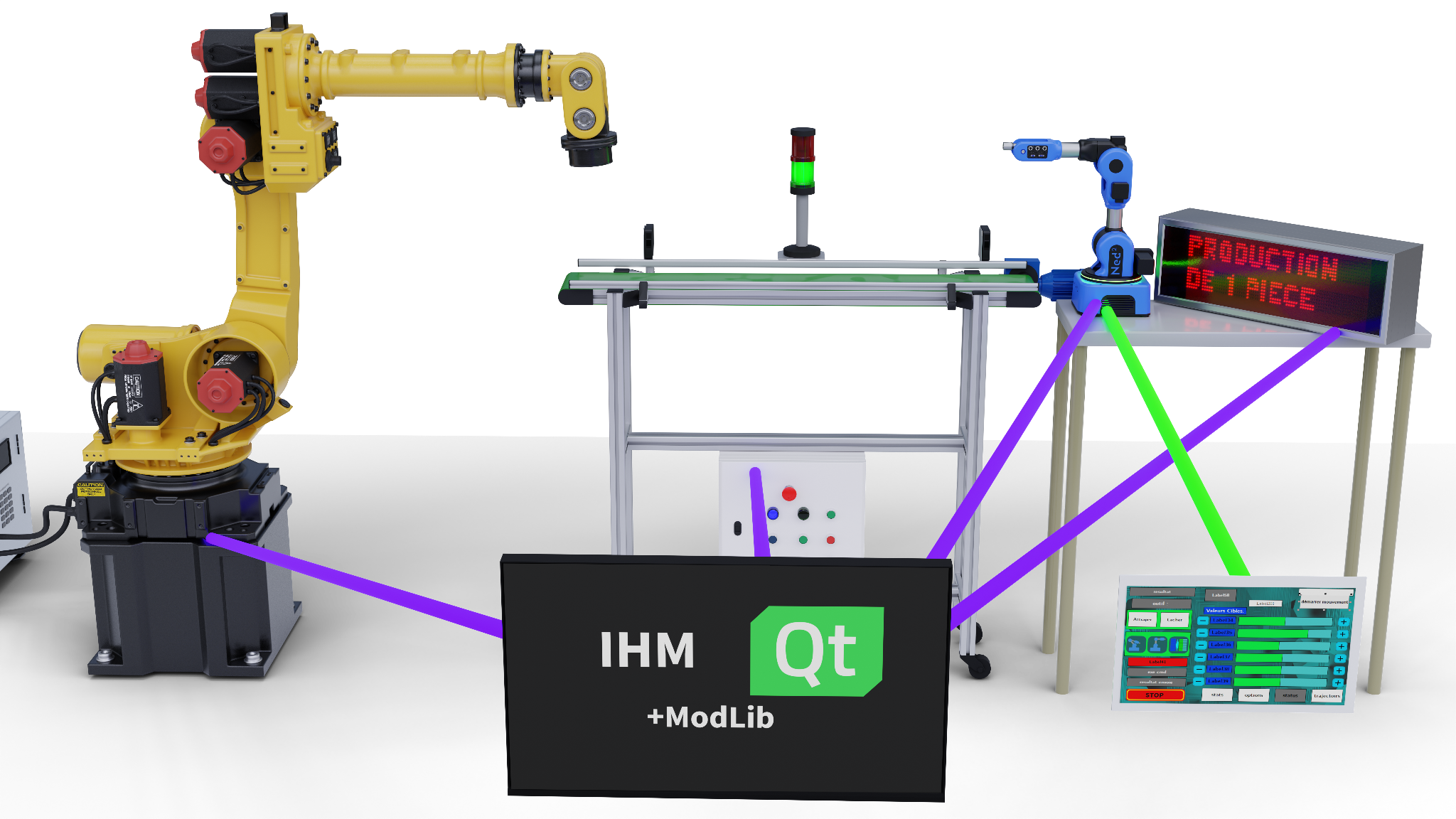
Pour gérer et superviser l'ensemble du système, une application graphique sera développée. Elle permet de contrôler et de surveiller tous les éléments du système. Grâce à cette interface, les opérateurs pourront facilement suivre le processus de manutention, ajuster les paramètres si nécessaire et garantir que tout fonctionne de manière optimale.

# 

# 1 - Objectif de réalisation dans ce projet

Mon objectif dans ce projet est de programmer et d'intégrer le robot FANUC afin qu'il puisse manipuler les pièces de manière précise et efficace. Pour cela, je vais également développer une bibliothèque de communication en Modbus/TCP, que j'ai nommée "ModLib". Cette bibliothèque assurera une interaction fluide et efficace entre tous les composants du système, permettant une coordination optimale des tâches de manutention. Grâce à ModLib, la communication entre le robot FANUC, le Niryo Ned2, et les autres éléments du système sera simplifiée et fiable, garantissant ainsi le bon fonctionnement de l'ensemble du processus automatisé.

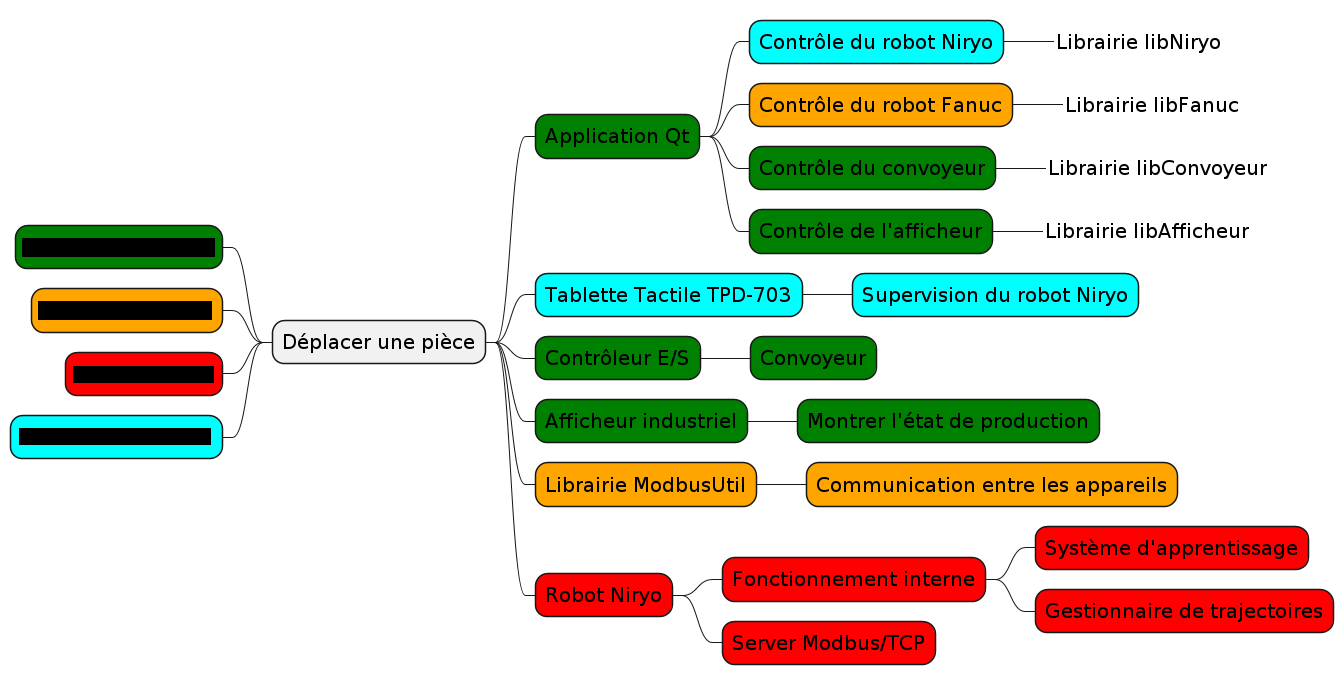
[[2]](#footnote-1)



#### Figure 1 : Vue 3D du Projet

# 2 - Présentation des tâches à réaliser

Pour illustrer clairement la répartition des tâches dans notre projet, nous avons créé un schéma coloré. Dans ce schéma, chaque couleur représente un ensemble de tâches spécifiques. Mes tâches sont en orange :



#### Figure 2 : Ensemble du projet

## 

## 

## 

## 2.1 - Tâches personnels

| **F3 : Contrôle – commande du robot 1 FANUC**   * Départ cycle, arrêt de la production * Synchronisation du superviseur avec ce robot par des flags DI/DO (In/Out) * Communication avec le protocole Modbus/TCP, fonction 23 * Gestion des signaux UOP (Panneau Opérateur Utilisateur) * Mise en condition de démarrage à distance : IMSTP, HOLD, SFSPB, ENBL * Sélection d'un programme sur le robot RSR, accusé de réception ACK * Démarrage du cycle : START * Surveillance de l'état du robot : PROGRUN, PAUSE, FAULT, etc. |
| --- |

| **F4 : Libraire pour le protocole Modbus/TCP, mode maître sur le superviseur**   * Pour la communication du superviseur vers les différents éléments de la PO * Classe ProtocoleModbus/TCP compilé sous forme de librairie dynamique * Ensemble des fonctions Modbus nécessaires à la communication avec les éléments matériels de la PO * Gestion des erreurs de communication |
| --- |

# 

# 3 - Situation dans le projet

Ce chapitre présente l’analyse du cahier des charges ainsi que la description de la

partie individuelle. Il présente aussi la mise en oeuvre de l’environnement de

développement nécessaire pour la réalisation de ce projet.

#### Figure 3 : Diagramme de Cas d’Utilisation

## 

## 3.1 - Analyse du cahier des charges

Le diagramme de déploiement (voir [Figure 3](#_c3a5ogukuz7b)) présente le système à produire.

**Supervision Globale de la Partie Opérative**

La supervision du système nécessite le développement d'une application graphique. Celle-ci doit être capable d'initialiser les matériels, de gérer le démarrage et l'arrêt des cycles de production, et de synchroniser les différents éléments du système (robot FANUC, convoyeur, robot Niryo). L'état de chaque élément doit être affiché de manière numérique ou graphique, et les données de production doivent être envoyées à un afficheur industriel.

**Contrôle-Commande du Robot FANUC**

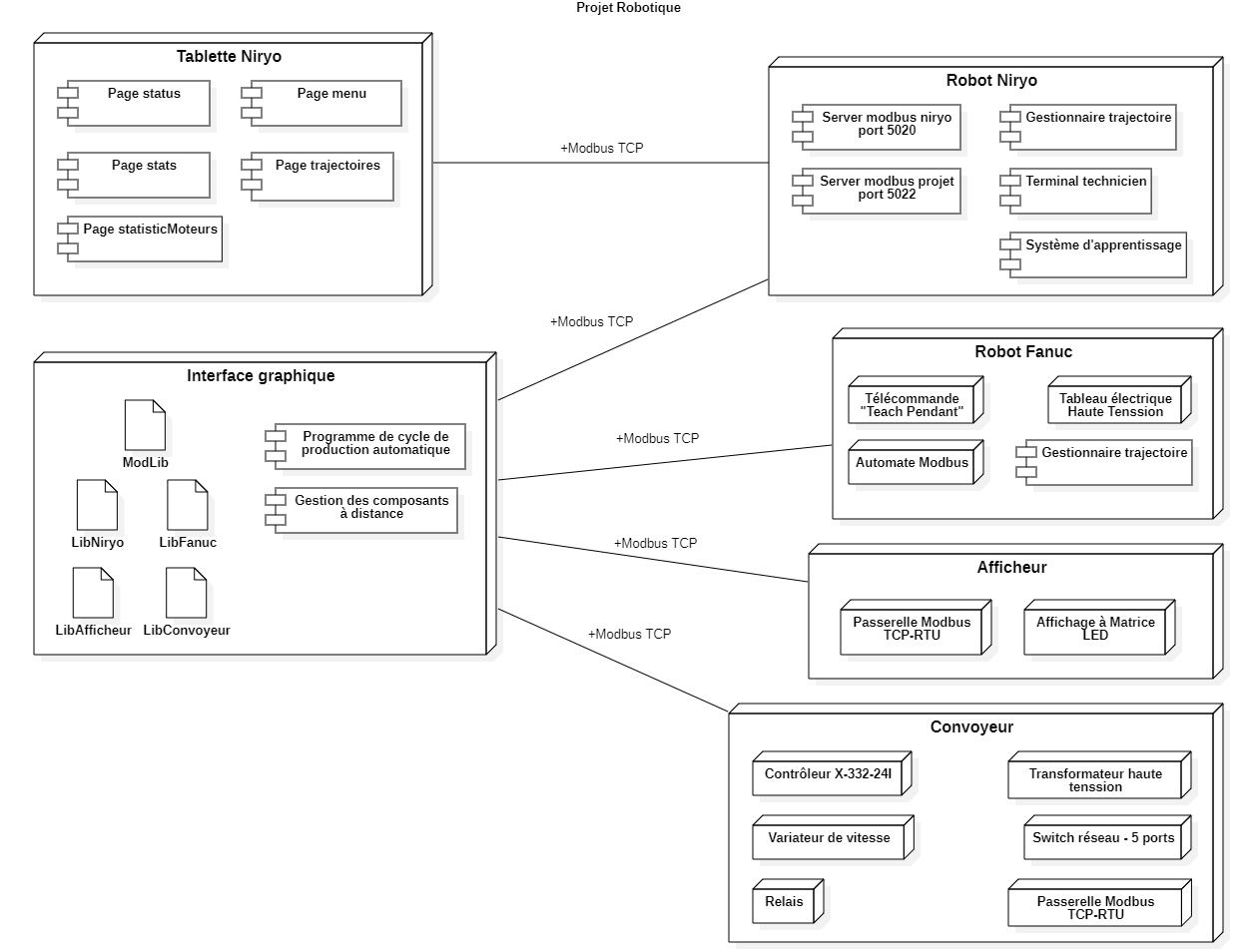
Le robot FANUC doit être intégré dans le cycle de production, avec des fonctions de départ et d'arrêt de cycle. La synchronisation avec le superviseur se fait par des flags d'entrée/sortie (DI/DO). La communication utilise le protocole Modbus/TCP et gère les signaux UOP (Panneau Opérateur Utilisateur) pour des conditions de démarrage à distance. La sélection et l'exécution de programmes se font via des commandes spécifiques, avec surveillance de l'état du robot (exécution, pause, erreur).

**Bibliothèque pour le Protocole Modbus/TCP**

Une bibliothèque pour le protocole Modbus/TCP est nécessaire pour la communication entre le superviseur et les différents éléments de la partie opérative. Cette bibliothèque doit inclure toutes les fonctions Modbus nécessaires et gérer les erreurs de communication.

**Affichage de l'État de la Production**

Un afficheur industriel est utilisé pour montrer l'état de la production en cours et les données de production. La communication se fait via le protocole Modbus/TCP.



#### Figure 4 : Diagramme de déploiement

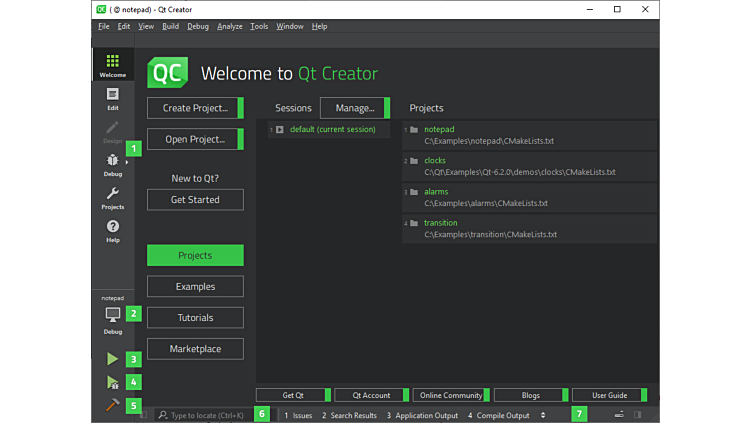
## 3.2 - Matériels et Logiciels

Pour réaliser les fonctions de notre projet de chaîne de manutention robotisée, j'ai dû préparer et configurer un environnement de développement adéquat. Cette préparation inclut l'installation de divers logiciels et outils nécessaires au développement, ainsi que la mise en place des librairies et frameworks spécifiques à notre projet. La création d'un environnement de développement robuste et adapté est essentielle pour garantir la réussite de notre projet en assurant la compatibilité, l'efficacité et la maintenabilité du code.

### 3.2.1 - QT Creator

Pour le projet de chaîne de manutention robotisée, je suis responsable des fonctions F3 (voir [2.1](#_abr1w68e7at7)) et F4 (voir [2.1](#_abr1w68e7at7)). Afin de réaliser ces tâches, je vais devoir installer et configurer plusieurs outils logiciels, notamment Qt Creator. Voici une explication détaillée de ce que cela implique et pourquoi j'ai choisi cet environnement de développement intégré (IDE[[3]](#footnote-2)).

Qt Creator est un environnement de développement intégré puissant et flexible qui est utilisé pour le développement d'applications multiplateformes. Pour ce projet, l'installation de Qt Creator est essentielle car elle permet de développer, tester et déployer les applications nécessaires pour le contrôle et la supervision des robots et du convoyeur.



#### Figure 5 : IDE QT Creator

Cet IDE permet de créer des applications multiplateformes (Windows, Linux, macOS) sans modifications majeures du code source, garantissant la compatibilité du système dans différents environnements.

Qt Creator est intégré avec Qt 6, facilitant le développement d'interfaces utilisateur modernes pour l'IHM de supervision globale (voir [Figure 1](#_t69oou9bzwf)). Il gère facilement les bibliothèques pour la communication Modbus/TCP, simplifiant la création des librairies dynamiques. Les outils avancés de développement, comme l'éditeur de code intelligent et le débogueur intégré, sont essentiels pour développer et déboguer efficacement les logiciels pour le robot FANUC.

De plus, Qt Creator dispose d'une documentation exhaustive et d'une large communauté de développeurs, ce qui aide à résoudre les problèmes rapidement. L'IDE encourage également les bonnes pratiques de codage et la documentation, assurant la qualité et la maintenabilité du code. En résumé, Qt Creator est idéal pour garantir un développement efficace et robuste des composants du projet.

### 3.2.2 - Télécommande “touch pendant”

La télécommande "Touch Pendant" joue un rôle crucial dans notre projet, spécialement conçu pour le contrôle et la supervision du robot industriel FANUC Mi10. Cet outil ergonomique permet une interaction intuitive avec le robot, facilitant ainsi la configuration des paramètres, les ajustements de mouvements en temps réel et la gestion des urgences grâce à ses arrêts d'urgence intégrés. Ses commandes précises améliorent non seulement l'efficacité opérationnelle mais aussi la sécurité des opérateurs.

**Activation des UOP (User Operator Panels) :**

| Accédez à [MENU] → Next → SYSTEM → Config.  7 : ENABLE UI SIGNALS = TRUE  45 : Remote/local setup : Remote  TYPES → Variables → 504 : $RMT\_MASTER = 0 |
| --- |

**Effectuez un reboot pour appliquer les modifications :**

| Appuyez sur [FCTN] → Next → Cycle Power → Yes pour redémarrer le système. |
| --- |

**Arrêt des programmes en cours :**

| Utilisez [FCTN] → Abort All pour interrompre tous les programmes en cours. |
| --- |

#### 

#### Figure 6 : Télécommande Fanuc Mi10

### 3.2.3 - Robot industriel FANUC MI10

Dans le cadre de notre projet de BTS, nous avons utilisé le bras robotique industriel FANUC Mi10 pour optimiser les performances de notre ligne de production automatisée. Ce choix s'appuie sur les nombreux avantages offerts par ce robot en termes de précision, de fiabilité et de compatibilité avec les systèmes de communication industriels.  
  
 Le FANUC Mi10 est conçu pour effectuer des tâches variées dans les environnements de production. Il se distingue par sa capacité à manipuler des objets avec une grande précision et rapidité, ce qui est essentiel pour les opérations nécessitant une manipulation délicate et répétitive. Le Mi10 est équipé de moteurs et de capteurs avancés qui assurent une précision de positionnement exceptionnelle, permettant ainsi d'atteindre une qualité constante dans la production.

**L’entreprise FANUC**

FANUC est une entreprise japonaise renommée, spécialisée dans les technologies d'automatisation industrielle. Fondée en 1956, elle est devenue un leader mondial dans la fabrication de robots industriels, de contrôleurs CNC (commande numérique par calculateur) et de systèmes de production automatisée. FANUC se distingue par son engagement envers l'innovation et la qualité, offrant des solutions robustes et fiables pour améliorer l'efficacité et la productivité des processus industriels. L'entreprise est reconnue pour ses produits de haute performance, sa recherche constante en développement technologique et son service client de premier ordre.

**Avantages de ce bras robotique**

L'un des principaux atouts du FANUC Mi10 est sa robustesse et sa fiabilité. Conçu pour fonctionner dans des environnements industriels exigeants, ce bras robotisé offre une performance durable et continue. Il est capable de fonctionner de manière autonome avec un minimum d'intervention humaine, ce qui réduit les risques d'erreur et améliore l'efficacité globale de la production. De plus, son interface utilisateur intuitive permet une programmation facile et rapide, ce qui réduit les besoins en formation des opérateurs et facilite l'adaptation aux changements de production.

Le Mi10 intègre également des fonctionnalités de sécurité avancées, telles que des capteurs de collision et des systèmes de surveillance en temps réel, garantissant un fonctionnement sûr et fiable. Ces caractéristiques permettent de réduire les temps d'arrêt et les coûts de maintenance, tout en augmentant la productivité.

Un autre avantage majeur du FANUC Mi10 est sa compatibilité avec le protocole de communication Modbus TCP. Modbus TCP est largement utilisé dans l'automatisation industrielle pour la transmission de données entre différents équipements. Cette compatibilité permet une intégration facile du Mi10 dans les réseaux industriels existants, facilitant ainsi la communication et la coordination entre différentes machines et systèmes. Grâce à cette compatibilité, il est possible de surveiller et de contrôler le robot à distance, optimisant ainsi les processus de production et la gestion des opérations industrielles.



### Figure 8 : Robot Fanuc MI10

Ce robot possède une grande portée, ce qui le rend particulièrement adapté à diverses applications industrielles. Voici les positions maximales qu'il peut atteindre :

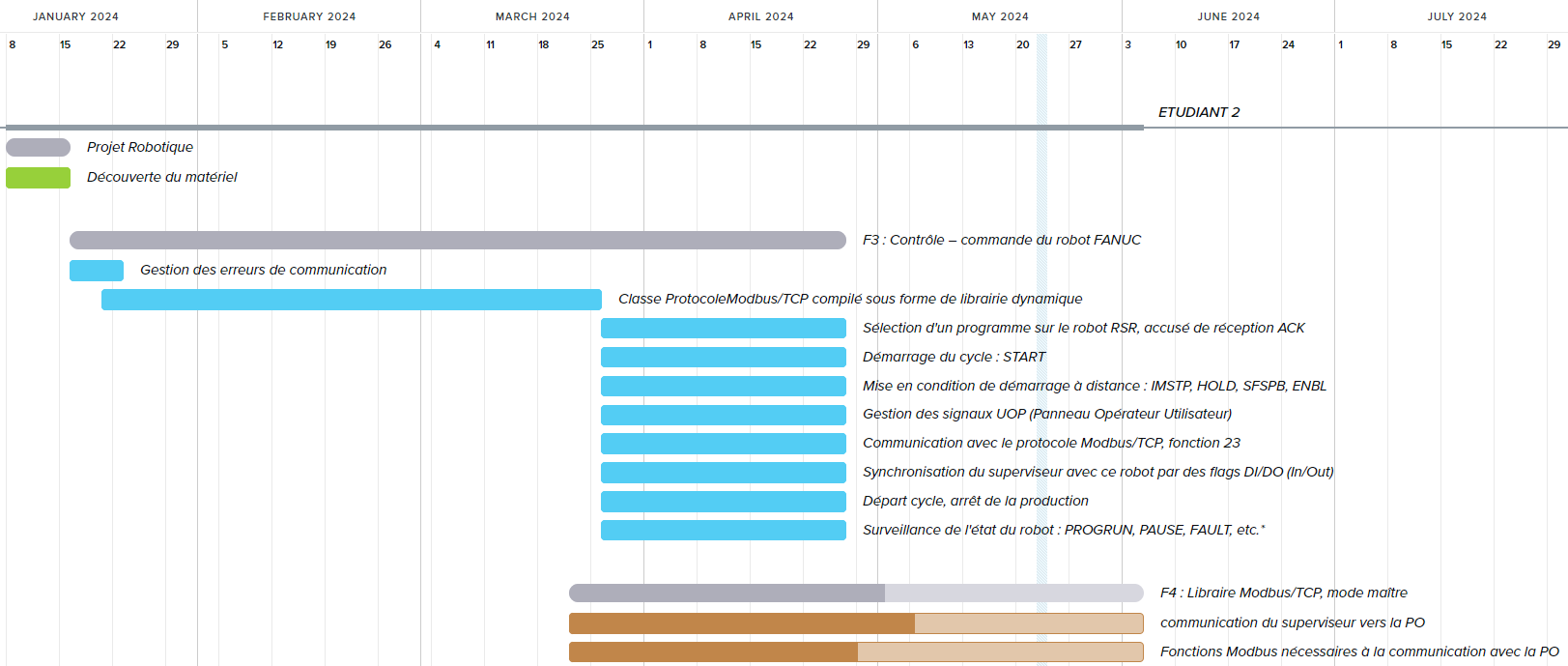
* Portée maximale horizontale : ~ 1420 mm (1,42 m)
* Portée maximale verticale : ~ 2520 mm (2,52 m)

## 3.4 - Planification des travaux

Afin de mener à bien ce projet, j’ai dû réaliser un diagramme de Gantt (voir [Figure 7](#_sta8s23euiy1)) pour

planifier les tâches que j’ai à faire. Ce diagramme regroupe les parties de

conception ainsi que de réalisation de ce projet.



#### Figure 7 : Diagramme de Gant

# 

# 4 - Conception du programme du Fanuc

Ce chapitre décrit la conception du programme de contrôle du Robot industriel FANUC MI10.

## 4.1 - Langage utilisé

Dans le cadre de mon projet personnel, j'ai utilisé le logiciel Qt Creator pour programmer le fonctionnement du robot FANUC Mi10. Qt Creator est un environnement de développement intégré basé sur le langage C++, qui est largement utilisé pour développer des applications multiplateformes avec une interface utilisateur graphique.

### 4.1.1 - Avantage du c++

**Puissance et Performance :**

Le C++ est reconnu pour sa performance et son efficacité, ce qui est crucial pour les applications robotiques qui nécessitent un traitement rapide et fiable des données.

**Flexibilité et Portabilité :**

Qt Creator permet de développer des applications qui peuvent être déployées sur différentes plateformes, ce qui est avantageux pour l'intégration et l'adaptation aux divers environnements industriels.

**Support et Documentation :**

C++ bénéficient d'une vaste communauté de développeurs et de nombreuses ressources de documentation. Cela facilite la résolution des problèmes et l'apprentissage continu, assurant un développement efficace et sans obstacles.

### 4.1.2 - Requête modbus

En premier lieu il est important de bien configurer son projet c’est pourquoi il faut installer les bibliothèques nécessaires :

**A glisser dans le fichier .pro de son projet :**

| QT += core gui serialbus  QT += core gui network |
| --- |

Ensuite il nous faut faire ne sorte de lire les uop et les écrires :

**Lire :**

| quint16*\** fanucBaseController::lireUOP()  {  statutModbusLecture = fanucModbus->lireNMots(10000, 2, motsDO);  *return* motsDO;  } | 1 : Définition d'une valeur initiale pour les UOP  2 :Modification des mots DI :  3 : Écriture des mots DI via Modbus : |
| --- | --- |

**Ecrire :**

| void fanucBaseController::ecrireUOPinit()  {  *//IMST+HOLD+SFSPD+ENBL*  quint16 initUOP=0x0087;  motsDI[0] = motsDI[0] | initUOP;    statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);  } | 1 : Définition et initialisation de la variable initUOP  2 : Mise à jour du tableau motsDI  3 : Écriture des mots DI via Modbus |
| --- | --- |

## 

## 4.2 - Arbre du projet Fanuc

Pour être sûr que le programme fonctionne bien et soit bien clair il est important de bien ranger son programme dans les bons fichiers :

# 

#### Figure 8 : Arbre du projet Fanuc

### 

En premier lieu la **fanucbasecontroller** s’occupe de gérer les étapes basses du projet, que ça soit la lecture et écriture d’uop ou de DI/DO. Ensuite j’ai mis en **placefanuctechnicianinterface** qui prend en compte l’interface graphique de mon projet de test et qui donne la possibilité à un technicien de faire une mise à jour simple au besoin (voir [Figure 9](#_js537le5shkd)). Pour terminer le fanucmain servira à l’avenir à prendre en compte toute les étapes haute du projet. Par exemple : Avancer, Gauche 30% et plus encore.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

#### Figure 9 : IHM graphique de test du projet FANUC

### 4.2.1 - UOP

Un UOP (User Operator Panel) est un ensemble de signaux standardisés utilisés pour interagir avec un système automatisé, tel qu'un robot industriel comme ceux produits par Fanuc. Ces signaux sont principalement destinés à faciliter la communication et le contrôle entre le robot et d'autres dispositifs, tels que des automates programmables d’où le choix de cette technologie dans le cahier des charges du projet pour le FANUC.

Ils jouent un rôle essentiel dans le contrôle, la sécurité et l'intégration des robots industriels comme ceux de Fanuc. Ils permettent à des dispositifs externes de commander et surveiller le robot, assurant ainsi une opération sécurisée et coordonnée dans les lignes de production. Ces signaux fournissent également des informations cruciales pour le diagnostic et la maintenance, tout en offrant une flexibilité d'adaptation aux besoins spécifiques des processus industriels. En résumé, les UOP facilitent une automatisation efficace tout en améliorant la fiabilité et la flexibilité des opérations industrielles.

# 

## 4.3 Problèmes rencontrés

**Configuration Initiale de la Communication Modbus/TCP**

L'un des principaux défis a été la configuration initiale de la communication via le protocole Modbus/TCP entre le robot FANUC et le système de contrôle. Comprendre les spécificités de ce protocole et assurer une intégration fluide avec les composants existants a demandé une recherche approfondie et des tests rigoureux. Grâce à une documentation détaillée et à une collaboration étroite avec les équipes techniques, nous avons pu surmonter ces défis et établir une communication robuste et fiable.

**Intégration des Signaux UOP pour le Démarrage à Distance**

Un autre défi significatif a été l'intégration des signaux UOP (Panneau Opérateur Utilisateur) pour permettre le démarrage à distance du robot FANUC. Ce processus impliquait la gestion des signaux IMSTP, HOLD, SFSPB, et ENBL, nécessitant une compréhension approfondie des spécifications techniques et une programmation précise pour assurer la sécurité et la fiabilité de l'opération. Après plusieurs itérations et tests sur banc d'essai, nous avons réussi à configurer ces signaux de manière efficace, permettant un contrôle précis du démarrage et de l'arrêt des cycles de production.

## 

## 

## 

## 

## 4.4 Diagramm de Classe du robot FANUC



#### Figure 10 : Diagramme de classe FANUC

# 

# 5 - Conception du programme de modlib

La bibliothèque modlib est une librairie dynamique développée spécifiquement pour faciliter la communication avec les systèmes utilisant le protocole Modbus/TCP, tel que le robot industriel FANUC MI10 dans notre projet. Cette bibliothèque est conçue pour simplifier l'accès aux fonctionnalités de communication Modbus et pour fournir une interface robuste entre le contrôleur et les dispositifs externes.

## 5.1 - Utilité et Avantages de modlib

**Simplification de la Communication Modbus**

Modlib simplifie grandement la mise en œuvre de la communication Modbus/TCP en encapsulant les détails techniques et en fournissant des méthodes conviviales pour lire et écrire des données. Cela réduit la complexité du code nécessaire pour gérer les échanges de données avec le robot FANUC, permettant ainsi un développement plus rapide et moins sujet aux erreurs.

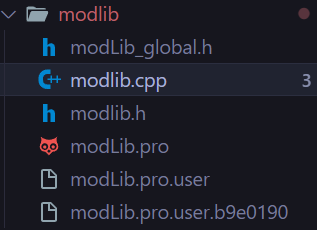
**Flexibilité et Adaptabilité**

Grâce à modlib, il est possible de configurer facilement les demandes et les réponses Modbus, tout en gérant les aspects spécifiques du protocole tels que la gestion des erreurs et la synchronisation des transactions. Cette flexibilité permet d'adapter la communication aux besoins spécifiques du projet et d'intégrer efficacement de nouvelles fonctionnalités ou des modifications ultérieures.

**Réutilisabilité et Maintenance Simplifiée**

En tant que bibliothèque dynamique, modlib favorise la réutilisabilité du code. Une fois intégrée et testée, elle peut être utilisée dans divers projets sans avoir à réécrire ou reproduire la logique de communication Modbus à chaque fois. De plus, cette approche simplifie la maintenance continue de la bibliothèque, assurant ainsi la pérennité et la fiabilité des systèmes qui l'utilisent.

## 5.2 - Arbre du projet modlib



#### 

#### 

#### 

#### 

#### Figure 11 : Arbre du projet modlib

## 

## 

## 5.3 - Problèmes rencontrés

**Gestion des exports et imports (Q\_DECL\_EXPORT et Q\_DECL\_IMPORT)**

Lors de la définition des classes et des fonctions dans ma bibliothèque, j'ai dû m'assurer qu'elles étaient correctement exportées pour être utilisées par d'autres applications. Cela implique de définir MODLIB\_EXPORT comme Q\_DECL\_EXPORT lorsque je comparais la bibliothèque et comme Q\_DECL\_IMPORT dans les applications clientes. La configuration correcte de ces directives conditionnelles (#if defined(MODLIB\_LIBRARY)) était cruciale pour éviter des erreurs de lien pendant la compilation.

| QT -= gui  QT += network  TEMPLATE = lib  TARGET = ModLib  CONFIG += c++17  DEFINES += MODLIB\_LIBRARY  SOURCES += \  modlib.cpp  HEADERS += \  modlib.h \  modLib\_global.h  unix {  target.path = /usr/lib  }  !isEmpty(target.path): INSTALLS += target |
| --- |

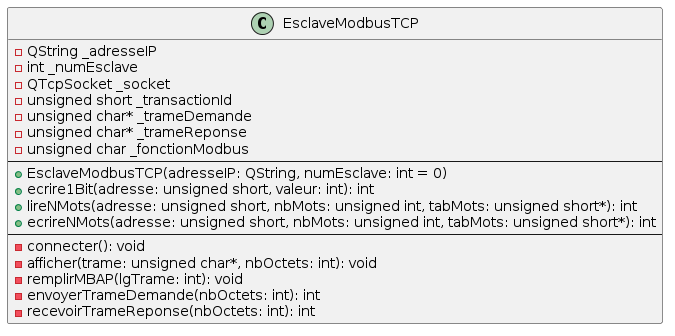
**Gestion des dépendances et inclusions de fichiers**

J'ai rencontré des problèmes d'inclusion de fichiers lorsque certains fichiers nécessaires à la bibliothèque n'étaient pas inclus correctement. Cela a provoqué des erreurs de compilation difficiles à résoudre. Pour résoudre cela, j'ai vérifié et rectifié les chemins d'inclusion dans mes fichiers d'en-tête (modlib.h, modlib\_global.h) et m'assurer que toutes les dépendances étaient correctement spécifiées.

| #include <QTcpSocket>  #include <QString>  #include <QTcpSocket>  #include <QString> |
| --- |

# 

## 5.4 - Diagramme de classe du projet modlib



#### Figure 12 : Diagramme de classe modlib

# 6 - Tests Unitaires FANUC

| **Méthode** | **Description** | **Entrées** | **Résultat attendu** | **Résultat obtenu (exemple)** |
| --- | --- | --- | --- | --- |
| **ecrire1Bit()** | Écriture réussie d'un bit | Adresse = 10, Valeur = 1 | REPONSE\_OK | Succès : Bit 10 écrit avec valeur 1 |
|  |  | Adresse = 10, Valeur = 0 (timeout simulé) | ERREUR\_TIMEOUT | Échec : Timeout lors de l'écriture |
| **lireNMots()** | Lecture réussie de plusieurs mots | Adresse = 20, NbMots = 5, TabMots non vide | REPONSE\_OK | Succès : Lecture de 5 mots à l'adresse 20 |
|  |  | Adresse = 20, NbMots = 5, TabMots non vide (timeout simulé) | ERREUR\_TIMEOUT | Échec : Timeout lors de la lecture |
| **ecrireNMots()** | Écriture réussie de plusieurs mots | Adresse = 30, NbMots = 5, TabMots avec valeurs | REPONSE\_OK | Succès : Écriture de 5 mots à l'adresse 30 |
|  |  | Adresse = 30, NbMots = 5, TabMots avec valeurs (timeout simulé) | ERREUR\_TIMEOUT | Échec : Timeout lors de l'écriture |
| **connecter()** | Connexion réussie au socket | Adresse IP valide | Connexion établie avec succès | Succès : Connexion établie avec IP 192.168.1.1 |
|  |  | Adresse IP invalide (timeout simulé) | ERREUR\_TIMEOUT | Échec : Timeout lors de la connexion |
| **afficher()** | Affichage correct d'une trame | Trame valide | Affichage correct de la trame hexadécimale | Affichage de la trame 0x01020304 |
| **remplirMBAP()** | Remplissage correct du MBAP | Longueur valide | MBAP correctement rempli | MBAP rempli avec ID transaction 123 |
| **envoyerTrameDemande()** | Envoi réussi d'une trame de demande | Taille de trame correcte | Trame envoyée avec succès | Succès : Trame de 12 octets envoyée |
| **recevoirTrameReponse()** | Réception réussie d'une trame de réponse | Taille de trame correcte | Trame reçue avec succès | Succès : Trame de 20 octets reçue |

# 7 - Test Unitaire modlib

# 

| **Méthode** | **Description** | **Entrées** | **Résultat attendu** | **Résultat obtenu (exemple)** |
| --- | --- | --- | --- | --- |
| ecrire1Bit() | Écriture réussie d'un bit | Adresse = 10, Valeur = 1 | REPONSE\_OK | Succès : Bit 10 écrit avec valeur 1 |
|  |  | Adresse = 10, Valeur = 0 (timeout simulé) | ERREUR\_TIMEOUT | Échec : Timeout lors de l'écriture |
| lireNMots() | Lecture réussie de plusieurs mots | Adresse = 20, NbMots = 5, TabMots non vide | REPONSE\_OK | Succès : Lecture de 5 mots à l'adresse 20 |
|  |  | Adresse = 20, NbMots = 5, TabMots non vide (timeout simulé) | ERREUR\_TIMEOUT | Échec : Timeout lors de la lecture |
| ecrireNMots() | Écriture réussie de plusieurs mots | Adresse = 30, NbMots = 5, TabMots avec valeurs | REPONSE\_OK | Succès : Écriture de 5 mots à l'adresse 30 |
|  |  | Adresse = 30, NbMots = 5, TabMots avec valeurs (timeout simulé) | ERREUR\_TIMEOUT | Échec : Timeout lors de l'écriture |
| connecter() | Connexion réussie au socket | Adresse IP valide | Connexion établie avec succès | Succès : Connexion établie avec IP 192.168.1.1 |
|  |  | Adresse IP invalide (timeout simulé) | ERREUR\_TIMEOUT | Échec : Timeout lors de la connexion |
| afficher() | Affichage correct d'une trame | Trame valide | Affichage correct de la trame hexadécimale | Affichage de la trame 0x01020304 |
| remplirMBAP() | Remplissage correct du MBAP | Longueur valide | MBAP correctement rempli | MBAP rempli avec ID transaction 123 |
| envoyerTrameDemande() | Envoi réussi d'une trame de demande | Taille de trame correcte | Trame envoyée avec succès | Succès : Trame de 12 octets envoyée |
| recevoirTrameReponse() | Réception réussie d'une trame de réponse | Taille de trame correcte | Trame reçue avec succès | Succès : Trame de 20 octets reçue |

# 

# Synthèse des tâches réalisées

# 

Ce chapitre permet de conclure ce rapport. Il présente les bilans technique et

personnel ainsi que les perspectives du projet.

## Bilan technique

Le développement et la conception de la bibliothèque ModLib et l'intégration du robot FANUC se sont déroulés de manière satisfaisante, répondant aux exigences du cahier des charges. Cette tâche a nécessité de nombreuses recherches pour maîtriser l'environnement de programmation du robot FANUC et le protocole Modbus/TCP. Des difficultés ont été rencontrées, notamment lors de la configuration initiale de la communication Modbus/TCP et l'intégration des signaux UOP. Ces obstacles ont été rapidement surmontés, permettant une mise en œuvre réussie.

Les tests de la bibliothèque ModLib ne sont pas détaillés dans ce rapport, bien qu'ils aient été réalisés avec succès, garantissant une communication fiable entre les composants du système.

## Bilan Personnel

Ce projet m'a permis d'acquérir de nombreuses compétences dans le domaine de l'automatisation industrielle. J'ai appris à programmer et intégrer un robot industriel, ainsi qu'à développer une bibliothèque de communication en Modbus/TCP. Les échanges avec mes collègues et mentors ont été précieux pour affiner et améliorer mon travail, rendant cette expérience très enrichissante.

## Perspective du projet

Le projet peut encore être amélioré. Par exemple, l'optimisation des performances du système pourrait être réalisée en utilisant un module “serialbus” donné par QT creator qui propose un très bon client. La gestion des trames modbus est bien plus performante que celle que j’ai pu créer et plus simple d’utilisation.

# Sitographie

**Site Fanuc :**

<https://www.fanuc.eu/fr/fr>

**Site QT :**

<https://doc.qt.io/>

**Chat GPT :**

<https://chatgpt.com/?oai-dm=1>

# Annexe

## Code du Fanuc :

**esclavemodbustcp.cpp**

#include "esclavemodbustcp.h"

#include "modlib"

EsclaveModbusTCP::EsclaveModbusTCP(QString adresseIP, int numEsclave){

\_adresseIP = adresseIP;

\_numEsclave = numEsclave;

\_transactionId = 1;

}

void EsclaveModbusTCP::remplirMBAP(int lgTrame){

*// identificateur de transaction*

\_trameDemande[0]=\_transactionId>>8&0xFF;

\_trameDemande[1]=\_transactionId&0xFF;

\_transactionId++;

*// Protocole Modbus*

\_trameDemande[2]=0;

\_trameDemande[3]=0;

*// Lg trame*

\_trameDemande[4]=(lgTrame-6)>>8 & 0xFF;

\_trameDemande[5]=(lgTrame-6) & 0xFF;

*// esclace*

\_trameDemande[6]=\_numEsclave;

}

int EsclaveModbusTCP::envoyerTrameDemande(int nbOctets){

connecter();

\_socket.write((char\*)\_trameDemande, nbOctets);

\_socket.flush();

*// Pour debug*

qDebug("Fonction : %d", \_fonctionModbus);

afficher(\_trameDemande, nbOctets);

*return* 0;

}

int EsclaveModbusTCP::recevoirTrameReponse(int nbOctets){

*/\**

*// Lecture sans traiter trame erreur Modbus*

*while (\_socket->bytesAvailable() < nbOctets) {*

*if (!\_socket->waitForReadyRead(TIMEOUT)) {*

*qDebug("ERREUR Timeout : waitForReadyRead");*

*return ERREUR\_TIMEOUT;*

*}*

*}*

*\_socket->read((char\*)\_trameReponse, nbOctets);*

*afficher(\_trameReponse, nbOctets);*

*\*/*

*// Lecture des 9 1er octets et vérification si trame d'erreur*

while (\_socket.bytesAvailable() < LG\_ERREUR) {

if (!\_socket.waitForReadyRead(TIMEOUT)) {

*//qDebug("ERREUR Timeout : waitForReadyRead");*

*return* ERREUR\_TIMEOUT;

}

}

\_socket.read((char\*)\_trameReponse, LG\_ERREUR);

*// Si trame d'erreur la traiter*

if(\_trameReponse[7] == \_fonctionModbus+128) {

qDebug("Réception d'une trame d'erreur");

afficher(\_trameReponse, LG\_ERREUR);

*return* \_trameReponse[8]\*-1;

}

*// Lire suite de la trame de reponse*

while (\_socket.bytesAvailable() < nbOctets-LG\_ERREUR) {

if (!\_socket.waitForReadyRead(TIMEOUT)) {

*//qDebug("ERREUR Timeout : waitForReadyRead");*

*return* ERREUR\_TIMEOUT;

}

}

\_socket.read((char\*)\_trameReponse+LG\_ERREUR, nbOctets-LG\_ERREUR);

afficher(\_trameReponse, nbOctets);

\_socket.disconnectFromHost();

*return* REPONSE\_OK;

}

int EsclaveModbusTCP::ecrire1Bit(unsigned short adresse, int valeur){

\_trameDemande = new unsigned char[LG\_DEMANDE\_ECRIRE1BIT];

\_trameReponse = new unsigned char[LG\_REPONSE\_ECRIRE1BIT];

\_fonctionModbus = FONCTION\_ECRIRE1BIT;

remplirMBAP(LG\_DEMANDE\_ECRIRE1BIT);

\_trameDemande[7]=\_fonctionModbus;

\_trameDemande[8] = adresse>>8 & 0x00FF;

\_trameDemande[9] = adresse & 0x00FF;

if(valeur>0) valeur=0xFF; else valeur=0;

\_trameDemande[10] = valeur;

\_trameDemande[11] = 0;

envoyerTrameDemande(LG\_DEMANDE\_ECRIRE1BIT);

int status;

status=recevoirTrameReponse(LG\_REPONSE\_ECRIRE1BIT);

delete[] \_trameDemande;

delete[] \_trameReponse;

*return* status;

}

int EsclaveModbusTCP::lireNMots(unsigned short adresse, unsigned int nbMots, unsigned short *\**tabMots) {

\_trameDemande = new unsigned char[LG\_DEMANDE\_LECTURENMOTS];

unsigned int nbOctetsReponse = LG\_REPONSE\_LECTURENMOTS+nbMots\*2;

\_trameReponse = new unsigned char[nbOctetsReponse];

\_fonctionModbus = FONCTION\_LECTURENMOTS;

remplirMBAP(LG\_DEMANDE\_LECTURENMOTS);

\_trameDemande[7] = \_fonctionModbus;

\_trameDemande[8] = adresse>>8 & 0x00FF;

\_trameDemande[9] = adresse & 0x00FF;

\_trameDemande[10] = nbMots>>8 & 0x00FF;

\_trameDemande[11] = nbMots & 0x00FF;

envoyerTrameDemande(LG\_DEMANDE\_LECTURENMOTS);

int status;

status=recevoirTrameReponse(nbOctetsReponse);

if(status==REPONSE\_OK){

for(unsigned int i=0; i<nbMots; i++){

tabMots[i]=static\_cast<quint16>(\_trameReponse[2\*i+9]<<8 | \_trameReponse[2\*i+10]);

}

}

delete[] \_trameDemande;

delete[] \_trameReponse;

*return* status;

}

int EsclaveModbusTCP::ecrireNMots(unsigned short adresse, unsigned int nbMots, unsigned short *\**tabMots) {

unsigned int nbOctetsDemande = LG\_DEMANDE\_ECRIRENMOTS+nbMots\*2;

\_trameDemande = new unsigned char[nbOctetsDemande];

\_trameReponse = new unsigned char[LG\_REPONSE\_ECRIRENMOTS];

\_fonctionModbus = FONCTION\_ECRIRENMOTS;

remplirMBAP(nbOctetsDemande);

\_trameDemande[7] = \_fonctionModbus;

\_trameDemande[8] = adresse>>8 & 0x00FF;

\_trameDemande[9] = adresse & 0x00FF;

\_trameDemande[10] = nbMots>>8 & 0x00FF;

\_trameDemande[11] = nbMots & 0x00FF;

\_trameDemande[12] = static\_cast<quint8>(nbMots\*2);

for(unsigned int i=0; i<nbMots; i++){

\_trameDemande[i\*2+13]=tabMots[i]>>8 & 0x00FF;

\_trameDemande[i\*2+14]=tabMots[i] & 0x00FF;

}

envoyerTrameDemande(nbOctetsDemande);

int status;

status=recevoirTrameReponse(LG\_REPONSE\_ECRIRENMOTS);

delete[] \_trameDemande;

delete[] \_trameReponse;

*return* status;

}

void EsclaveModbusTCP::connecter()

{

\_socket.connectToHost(\_adresseIP, PORT\_MODBUS\_TCP);

\_socket.waitForConnected(2000);

if(\_socket.state() != QAbstractSocket::ConnectedState){

qDebug("Non connecte");

\_socket.close();

}

else qDebug("connection OK");

}

void EsclaveModbusTCP::afficher(unsigned char *\**trame, int nbOctets){

QByteArray tab((*const* char\*)trame, nbOctets);

qDebug("%d octets : %s", nbOctets, QString(tab.toHex()).toLocal8Bit().constData() );

}

**esclavemodbustcp.h**

**#ifndef ESCLAVEMODBUSTCP\_H**

**#define ESCLAVEMODBUSTCP\_H**

**#include<QTcpSocket>**

***// LG pour LONGUEUR***

***const* unsigned int PORT\_MODBUS\_TCP = 502;**

***const* unsigned int LG\_MBAP = 7;**

***const* unsigned int FONCTION\_ECRIRE1BIT = 5;**

***const* unsigned int LG\_DEMANDE\_ECRIRE1BIT = LG\_MBAP+5;**

***const* unsigned int LG\_REPONSE\_ECRIRE1BIT = LG\_MBAP+5;**

***const* unsigned char FONCTION\_LECTURENMOTS = 3;**

***const* unsigned int LG\_DEMANDE\_LECTURENMOTS = LG\_MBAP+5;**

***const* unsigned int LG\_REPONSE\_LECTURENMOTS = LG\_MBAP+2;**

***const* unsigned char FONCTION\_ECRIRENMOTS = 16;**

***const* unsigned int LG\_DEMANDE\_ECRIRENMOTS = LG\_MBAP+6;**

***const* unsigned int LG\_REPONSE\_ECRIRENMOTS = LG\_MBAP+5;**

***const* unsigned int REPONSE\_OK=0;**

***const* unsigned int TIMEOUT=2000;**

***const* unsigned int ERREUR\_TIMEOUT=-257;**

***const* unsigned int LG\_ERREUR=LG\_MBAP+2;**

**class EsclaveModbusTCP {**

**private:**

**QString \_adresseIP;**

**int \_numEsclave;**

**QTcpSocket \_socket;**

**unsigned short \_transactionId;**

**unsigned char\* \_trameDemande;**

**unsigned char\* \_trameReponse;**

**unsigned char \_fonctionModbus;**

**void connecter();**

**void afficher (unsigned char*\** trame, int nbOctets);**

**void remplirMBAP (int lgTrame);**

**int envoyerTrameDemande (int nbOctets);**

**int recevoirTrameReponse(int nbOctets);**

**public:**

**EsclaveModbusTCP(QString adresseIP, int numEsclave=0);**

**int ecrire1Bit (unsigned short adresse, int valeur);**

**int lireNMots (unsigned short adresse, unsigned int nbMots, unsigned short*\** tabMots);**

**int ecrireNMots (unsigned short adresse, unsigned int nbMots, unsigned short*\** tabMots);**

**};**

**#endif *// ESCLAVEMODBUSTCP\_H***

**fanucbasecontroller.cpp**

**#include "fanucbasecontroller.h"**

**fanucBaseController::fanucBaseController() {**

***//robotMi10 = new EsclaveModbusTCP("172.20.81.100",35);***

**fanucModbus = new EsclaveModbusTCP("192.168.1.11");**

**}**

**fanucBaseController::~fanucBaseController()**

**{**

**delete fanucModbus;**

**}**

**void fanucBaseController::ecrireDI(int num, bool etat)**

**{**

***// DI115=bit2 du mot 0x1***

**quint16 bitDI=1<<(num-113);**

**if(etat==true) {**

**motsDI[1]=motsDI[1] | bitDI;**

**} else {**

**bitDI =~bitDI;**

**motsDI[1]=motsDI[1] & bitDI;**

**}**

**statutModbusEcriture = fanucModbus->ecrireNMots(0,2,motsDI);**

**}**

**quint16*\** fanucBaseController::lireUOP()**

**{**

**statutModbusLecture = fanucModbus->lireNMots(10000, 2, motsDO);**

***return* motsDO;**

**}**

**void fanucBaseController::resetUOP()**

**{**

**motsDI[0] = 0;**

**motsDI[1] = 0;**

***//statutModbusEcriture = robotMi10->ecrireNMots(100,1,motsDI);***

**statutModbusEcriture = fanucModbus->ecrireNMots(0,2,motsDI);**

**}**

**void fanucBaseController::ecrireUOPinit()**

**{**

***//IMST+HOLD+SFSPD+ENBL***

**quint16 initUOP=0x0087;**

**motsDI[0] = motsDI[0] | initUOP;**

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**}**

**void fanucBaseController::ecrireFaultReset()**

**{**

***// FaultReset***

**quint16 faultResetUOP=0x0010;**

**motsDI[0] = motsDI[0] | faultResetUOP;**

***//statutModbusEcriture = robotMi10->ecrireNMots(100,1, motsDI);***

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**QThread::sleep(1);**

**faultResetUOP=~faultResetUOP; *// Complément à 1***

**motsDI[0] = motsDI[0] & faultResetUOP;**

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**}**

**void fanucBaseController::ecrireSTART()**

**{**

***//quint16 bitStart=0x1<<5;***

**quint16 bitStart=0x0020;**

**motsDI[0] = motsDI[0] | bitStart;**

**statutModbusEcriture = fanucModbus->ecrireNMots(0,2,motsDI);**

**QThread::sleep(1);**

**bitStart =~ bitStart;**

**motsDI[0] = motsDI[0] & bitStart;**

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**}**

**void fanucBaseController::ecrireHOLD(bool etat)**

**{**

**quint16 bitHOLD = 0x0002;**

***// true = HOLD à ON***

**if (etat==true){**

**motsDI[0] = motsDI[0] | bitHOLD;**

**} else {**

**motsDI[0] = motsDI[0] & (~bitHOLD);**

**}**

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**}**

**void fanucBaseController::ecrireRSR(int rsr)**

**{**

***//RSR***

**quint16 initUOP=0x1<<(rsr+7);**

**motsDI[0] = (motsDI[0] & 0x00FF) | initUOP;**

***//statutModbusEcriture = robotMi10->ecrireNMots(100,1, motsDI);***

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**QThread::sleep(1);**

**motsDI[0] = motsDI[0] & 0x00FF;**

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**}**

**void fanucBaseController::ecrireCSTOPI()**

**{**

**quint16 bitCSTOPI=0x0008;**

**motsDI[0] = motsDI[0] | bitCSTOPI;**

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**QThread::sleep(1);**

**motsDI[0] = motsDI[0] & (~bitCSTOPI);**

**statutModbusEcriture = fanucModbus->ecrireNMots(0, 2, motsDI);**

**}**

**fanucbasecontroller.h**

**#ifndef FANUCBASECONTROLLER\_H**

**#define FANUCBASECONTROLLER\_H**

**#include <QString>**

**#include <QTimer>**

**#include <QThread>**

**#include "modlib.h"**

**class fanucBaseController**

**{**

**public:**

**fanucBaseController();**

**~fanucBaseController();**

**quint16*\** lireUOP();**

**void resetUOP();**

**void ecrireUOPinit();**

**void ecrireFaultReset();**

**void ecrireSTART();**

**void ecrireHOLD(bool etat);**

**void ecrireRSR(int rsr);**

**void ecrireCSTOPI();**

**void ecrireDI(int num, bool etat);**

**quint16 motsDI[2]={0,0};**

**quint16 motsDO[2];**

**int statutModbusLecture=-1;**

**int statutModbusEcriture=-1;**

**protected:**

**EsclaveModbusTCP\* fanucModbus;**

**};**

**#endif *// FANUCBASECONTROLLER\_H***

**fanucmain.cpp**

**#include "fanucmain.h"**

**fanucMain::fanucMain() {**

**}**

**fanuctechnitianinterface.cpp**

**#include "fanuctechnitianinterface.h"**

**#include "ui\_fanuctechnitianinterface.h"**

**FanucTechnitianInterface::FanucTechnitianInterface(QWidget \*parent)**

**: QMainWindow(parent)**

**, ui(new Ui::FanucTechnitianInterface)**

**{**

**ui->setupUi(this);**

**ui->spinBox->setMinimum(1);**

**ui->spinBox->setMaximum(8);**

**ui->spinBox->setValue(8);**

**fanuc = new fanucBaseController();**

**afficherMotsDI();**

**timerLire.setInterval(800);**

***//timerLire.start();***

**connect(ui->pushButton\_quitter,&QPushButton::clicked, this, &FanucTechnitianInterface::close);**

**connect(&timerLire, &QTimer::timeout, this, &FanucTechnitianInterface::lireUOP);**

**ui->pushButton\_stopLecture->setCheckable(true);**

**ui->pushButton\_stopLecture->toggle();**

**connect(ui->pushButton\_stopLecture, &QPushButton::toggled, this, &FanucTechnitianInterface::startStopLireUOP);**

**connect(ui->pushButton\_resetUOP, &QPushButton::clicked, this, &FanucTechnitianInterface::resetUOP);**

**connect(ui->pushButton\_initUOP, &QPushButton::clicked, this, &FanucTechnitianInterface::ecrireUOPinit);**

**connect(ui->pushButton\_faultReset, &QPushButton::clicked, this, &FanucTechnitianInterface::ecrireFaultReset);**

**connect(ui->pushButton\_RSR, &QPushButton::clicked, this, &FanucTechnitianInterface::ecrireRSR);**

**ui->pushButton\_appliquerDI1->setCheckable(true);**

**connect(ui->pushButton\_appliquerDI1, &QPushButton::toggled, this, &FanucTechnitianInterface::ecrireDI1);**

**ui->pushButton\_appliquerDI2->setCheckable(true);**

**connect(ui->pushButton\_appliquerDI2, &QPushButton::toggled, this, &FanucTechnitianInterface::ecrireDI2);**

**connect(ui->pushButton\_START,&QPushButton::clicked, this, &FanucTechnitianInterface::ecrireSTART );**

**ui->pushButton\_HOLD->setCheckable(true);**

**connect(ui->pushButton\_HOLD, &QPushButton::toggled, this, &FanucTechnitianInterface::ecrireHOLD);**

**connect(ui->pushButton\_CSTOPI, &QPushButton::clicked, this, &FanucTechnitianInterface::ecrireCSTOPI);**

**}**

**FanucTechnitianInterface::~FanucTechnitianInterface()**

**{**

**delete fanuc;**

**delete ui;**

**}**

**void FanucTechnitianInterface::afficherStatutModbus()**

**{**

**ui->label\_statusModbusLecture->setText(QString::number(fanuc->statutModbusLecture));**

**ui->label\_statusModbusEcriture->setText(QString::number(fanuc->statutModbusEcriture));**

**}**

**void FanucTechnitianInterface::afficherMotsDI()**

**{**

**QString svaleur;**

**svaleur = QString::number(fanuc->motsDI[0], 16);**

**ui->label\_mot0hex->setText(svaleur.toUpper());**

**svaleur = QString::number(fanuc->motsDI[0], 2);**

**ui->label\_mot0bin->setText(svaleur);**

**svaleur = QString::number(fanuc->motsDI[1], 16);**

**ui->label\_mot0bin->setText(svaleur.toUpper());**

**svaleur = QString::number(fanuc->motsDI[1], 2);**

**ui->label\_mot1bin->setText(svaleur);**

**}**

**void FanucTechnitianInterface::lireUOP()**

**{**

**quint16\* mots=fanuc->lireUOP();**

**QString svaleur;**

**afficherStatutModbus();**

**svaleur = QString::number(mots[0], 16);**

**ui->label\_mot10000hex->setText(svaleur.toUpper());**

**svaleur = QString::number(mots[0], 2);**

**ui->label\_mot10000bin->setText(svaleur);**

**svaleur="";**

**for(int i=0; i<16; i++){**

**if( (mots[0]& (1<<i)) != 0)**

**svaleur=svaleur + labelDO0[i] +" - ";**

**}**

**ui->label\_mot10000label->setText(svaleur);**

**svaleur = QString::number(mots[1], 16);**

**ui->label\_mot10001hex->setText(svaleur.toUpper());**

**svaleur = QString::number(mots[1], 2);**

**ui->label\_mot10001bin->setText(svaleur);**

**svaleur="";**

**for(int i=0; i<16; i++){**

**if( (mots[1]& (1<<i)) != 0)**

**svaleur=svaleur + labelDO1[i] +" - ";**

**}**

**ui->label\_mot10001label->setText(svaleur);**

**}**

**void FanucTechnitianInterface::startStopLireUOP(bool etat)**

**{**

**if(etat==true) timerLire.stop();**

**else timerLire.start();**

**}**

**void FanucTechnitianInterface::resetUOP()**

**{**

**fanuc->motsDI[0] = 0;**

**fanuc->motsDI[1] = 0;**

**afficherMotsDI();**

**fanuc->resetUOP();**

**afficherStatutModbus();**

**}**

**void FanucTechnitianInterface::ecrireUOPinit()**

**{**

**fanuc->ecrireUOPinit();**

**afficherMotsDI();**

**afficherStatutModbus();**

**}**

**void FanucTechnitianInterface::ecrireFaultReset()**

**{**

**fanuc->ecrireFaultReset();**

**afficherMotsDI();**

**afficherStatutModbus();**

**}**

**void FanucTechnitianInterface::ecrireSTART()**

**{**

**fanuc->ecrireSTART();**

**afficherMotsDI();**

**afficherStatutModbus();**

**}**

**void FanucTechnitianInterface::ecrireHOLD(bool etat)**

**{**

**fanuc->ecrireHOLD(etat);**

**afficherMotsDI();**

**afficherStatutModbus();**

**}**

**void FanucTechnitianInterface::ecrireRSR()**

**{**

**fanuc->ecrireRSR(ui->spinBox->value());**

**afficherMotsDI();**

**afficherStatutModbus();**

**}**

**void FanucTechnitianInterface::ecrireDI1(bool etat)**

**{**

**int numDI=ui->spinBox\_DI1->value();**

**fanuc->ecrireDI(numDI, etat);**

**afficherMotsDI();**

**afficherStatutModbus();**

**}**

**void FanucTechnitianInterface::ecrireDI2(bool etat)**

**{**

**int numDI=ui->spinBox\_DI2->value();**

**fanuc->ecrireDI(numDI, etat);**

**afficherMotsDI();**

**afficherStatutModbus();**

**}**

**void FanucTechnitianInterface::ecrireCSTOPI()**

**{**

**fanuc->ecrireCSTOPI();**

**afficherMotsDI();**

**afficherStatutModbus();**

**}**

**fanuctechnitianinterface.h**

**#ifndef FANUCTECHNITIANINTERFACE\_H**

**#define FANUCTECHNITIANINTERFACE\_H**

**#include <QMainWindow>**

**#include <QTimer>**

**#include <QThread>**

**#include "fanucbasecontroller.h"**

**QT\_BEGIN\_NAMESPACE**

**namespace Ui { class FanucTechnitianInterface; }**

**QT\_END\_NAMESPACE**

**class FanucTechnitianInterface : public QMainWindow**

**{**

**Q\_OBJECT**

**public:**

**FanucTechnitianInterface(QWidget *\**parent = nullptr);**

**~FanucTechnitianInterface();**

**private:**

**Ui::FanucTechnitianInterface \*ui;**

**fanucBaseController \*fanuc;**

**QTimer timerLire;**

**QString labelDO0[16]={"CMDENBL","SYSRDY","PROGRAM","PAUSED","HELD","FAULT","ATPERCH","TPENBL","BATALM","BUSY","ACK1","ACK2","ACK3","ACK4","ACK5","ACK6"};**

**QString labelDO1[16]={"ACK7","ACK8","SNACK","RESERVED","DO133","DO134","DO135","DO136","DO137","DO138","DO139","DO140","DO141","DO142","DO143","DO144"};**

**void afficherStatutModbus();**

**void afficherMotsDI();**

**private slots:**

**void lireUOP();**

**void startStopLireUOP(bool etat);**

**void resetUOP();**

**void ecrireUOPinit();**

**void ecrireFaultReset();**

**void ecrireSTART();**

**void ecrireHOLD(bool etat);**

**void ecrireRSR();**

**void ecrireDI1(bool etat);**

**void ecrireDI2(bool etat);**

**void ecrireCSTOPI();**

**};**

**#endif *// FANUCTECHNITIANINTERFACE\_H***

**main.cpp**

**#include "fanuctechnitianinterface.h"**

**#include <QApplication>**

**int main(int argc, char *\**argv[])**

**{**

**QApplication a(argc, argv);**

**FanucTechnitianInterface w;**

**w.show();**

***return* a.exec();**

**}**

## 

## 

## Code du modlib :

**modLib\_global.h**

**#ifndef MODLIB\_GLOBAL\_H**

**#define MODLIB\_GLOBAL\_H**

**#include <QTcpSocket>**

**#include <QString>**

**#if defined(MODLIB\_LIBRARY)**

**# define MODLIB\_EXPORT Q\_DECL\_EXPORT**

**#else**

**# define MODLIB\_EXPORT Q\_DECL\_IMPORT**

**#endif**

**#endif *// MODLIB\_GLOBAL\_H***

**modlib.cpp**

**#include "modlib.h"**

**#include <QDebug>**

**EsclaveModbusTCP::EsclaveModbusTCP(QString adresseIP, int numEsclave){**

**\_adresseIP = adresseIP;**

**\_numEsclave = numEsclave;**

**\_transactionId = 1;**

**}**

**void EsclaveModbusTCP::remplirMBAP(int lgTrame){**

***// identificateur de transaction***

**\_trameDemande[0]=\_transactionId>>8&0xFF;**

**\_trameDemande[1]=\_transactionId&0xFF;**

**\_transactionId++;**

***// Protocole Modbus***

**\_trameDemande[2]=0;**

**\_trameDemande[3]=0;**

***// Lg trame***

**\_trameDemande[4]=(lgTrame-6)>>8 & 0xFF;**

**\_trameDemande[5]=(lgTrame-6) & 0xFF;**

***// esclace***

**\_trameDemande[6]=\_numEsclave;**

**}**

**int EsclaveModbusTCP::envoyerTrameDemande(int nbOctets){**

**connecter();**

**\_socket.write((char\*)\_trameDemande, nbOctets);**

**\_socket.flush();**

***// Pour debug***

**qDebug("Fonction : %d", \_fonctionModbus);**

**afficher(\_trameDemande, nbOctets);**

***return* 0;**

**}**

**int EsclaveModbusTCP::recevoirTrameReponse(int nbOctets){**

***/\****

***// Lecture sans traiter trame erreur Modbus***

***while (\_socket->bytesAvailable() < nbOctets) {***

***if (!\_socket->waitForReadyRead(TIMEOUT)) {***

***qDebug("ERREUR Timeout : waitForReadyRead");***

***return ERREUR\_TIMEOUT;***

***}***

***}***

***\_socket->read((char\*)\_trameReponse, nbOctets);***

***afficher(\_trameReponse, nbOctets);***

***\*/***

***// Lecture des 9 1er octets et vérification si trame d'erreur***

**while (\_socket.bytesAvailable() < LG\_ERREUR) {**

**if (!\_socket.waitForReadyRead(TIMEOUT)) {**

***//qDebug("ERREUR Timeout : waitForReadyRead");***

***return* ERREUR\_TIMEOUT;**

**}**

**}**

**\_socket.read((char\*)\_trameReponse, LG\_ERREUR);**

***// Si trame d'erreur la traiter***

**if(\_trameReponse[7] == \_fonctionModbus+128) {**

**qDebug("Réception d'une trame d'erreur");**

**afficher(\_trameReponse, LG\_ERREUR);**

***return* \_trameReponse[8]\*-1;**

**}**

***// Lire suite de la trame de reponse***

**while (\_socket.bytesAvailable() < nbOctets-LG\_ERREUR) {**

**if (!\_socket.waitForReadyRead(TIMEOUT)) {**

***//qDebug("ERREUR Timeout : waitForReadyRead");***

***return* ERREUR\_TIMEOUT;**

**}**

**}**

**\_socket.read((char\*)\_trameReponse+LG\_ERREUR, nbOctets-LG\_ERREUR);**

**afficher(\_trameReponse, nbOctets);**

**\_socket.disconnectFromHost();**

***return* REPONSE\_OK;**

**}**

**int EsclaveModbusTCP::ecrire1Bit(unsigned short adresse, int valeur){**

**\_trameDemande = new unsigned char[LG\_DEMANDE\_ECRIRE1BIT];**

**\_trameReponse = new unsigned char[LG\_REPONSE\_ECRIRE1BIT];**

**\_fonctionModbus = FONCTION\_ECRIRE1BIT;**

**remplirMBAP(LG\_DEMANDE\_ECRIRE1BIT);**

**\_trameDemande[7]=\_fonctionModbus;**

**\_trameDemande[8] = adresse>>8 & 0x00FF;**

**\_trameDemande[9] = adresse & 0x00FF;**

**if(valeur>0) valeur=0xFF; else valeur=0;**

**\_trameDemande[10] = valeur;**

**\_trameDemande[11] = 0;**

**envoyerTrameDemande(LG\_DEMANDE\_ECRIRE1BIT);**

**int status;**

**status=recevoirTrameReponse(LG\_REPONSE\_ECRIRE1BIT);**

**delete[] \_trameDemande;**

**delete[] \_trameReponse;**

***return* status;**

**}**

**int EsclaveModbusTCP::lireNMots(unsigned short adresse, unsigned int nbMots, unsigned short *\**tabMots) {**

**\_trameDemande = new unsigned char[LG\_DEMANDE\_LECTURENMOTS];**

**unsigned int nbOctetsReponse = LG\_REPONSE\_LECTURENMOTS+nbMots\*2;**

**\_trameReponse = new unsigned char[nbOctetsReponse];**

**\_fonctionModbus = FONCTION\_LECTURENMOTS;**

**remplirMBAP(LG\_DEMANDE\_LECTURENMOTS);**

**\_trameDemande[7] = \_fonctionModbus;**

**\_trameDemande[8] = adresse>>8 & 0x00FF;**

**\_trameDemande[9] = adresse & 0x00FF;**

**\_trameDemande[10] = nbMots>>8 & 0x00FF;**

**\_trameDemande[11] = nbMots & 0x00FF;**

**envoyerTrameDemande(LG\_DEMANDE\_LECTURENMOTS);**

**int status;**

**status=recevoirTrameReponse(nbOctetsReponse);**

**if(status==REPONSE\_OK){**

**for(unsigned int i=0; i<nbMots; i++){**

**tabMots[i]=static\_cast<quint16>(\_trameReponse[2\*i+9]<<8 | \_trameReponse[2\*i+10]);**

**}**

**}**

**delete[] \_trameDemande;**

**delete[] \_trameReponse;**

***return* status;**

**}**

**int EsclaveModbusTCP::ecrireNMots(unsigned short adresse, unsigned int nbMots, unsigned short *\**tabMots) {**

**unsigned int nbOctetsDemande = LG\_DEMANDE\_ECRIRENMOTS+nbMots\*2;**

**\_trameDemande = new unsigned char[nbOctetsDemande];**

**\_trameReponse = new unsigned char[LG\_REPONSE\_ECRIRENMOTS];**

**\_fonctionModbus = FONCTION\_ECRIRENMOTS;**

**remplirMBAP(nbOctetsDemande);**

**\_trameDemande[7] = \_fonctionModbus;**

**\_trameDemande[8] = adresse>>8 & 0x00FF;**

**\_trameDemande[9] = adresse & 0x00FF;**

**\_trameDemande[10] = nbMots>>8 & 0x00FF;**

**\_trameDemande[11] = nbMots & 0x00FF;**

**\_trameDemande[12] = static\_cast<quint8>(nbMots\*2);**

**for(unsigned int i=0; i<nbMots; i++){**

**\_trameDemande[i\*2+13]=tabMots[i]>>8 & 0x00FF;**

**\_trameDemande[i\*2+14]=tabMots[i] & 0x00FF;**

**}**

**envoyerTrameDemande(nbOctetsDemande);**

**int status;**

**status=recevoirTrameReponse(LG\_REPONSE\_ECRIRENMOTS);**

**delete[] \_trameDemande;**

**delete[] \_trameReponse;**

***return* status;**

**}**

**void EsclaveModbusTCP::connecter()**

**{**

**\_socket.connectToHost(\_adresseIP, PORT\_MODBUS\_TCP);**

**\_socket.waitForConnected(2000);**

**if(\_socket.state() != QAbstractSocket::ConnectedState){**

**qDebug("Non connecte");**

**\_socket.close();**

**exit(1);**

**}**

**else qDebug("connection OK");**

**}**

**void EsclaveModbusTCP::afficher(unsigned char *\**trame, int nbOctets){**

**QByteArray tab((*const* char\*)trame, nbOctets);**

**qDebug("%d octets : %s", nbOctets, QString(tab.toHex()).toLocal8Bit().constData() );**

**}**

**modlib.h**

**#ifndef ESCLAVEMODBUSTCP\_H**

**#define ESCLAVEMODBUSTCP\_H**

**#include <QTcpSocket>**

**#include <QString>**

***// LG pour LONGUEUR***

***const* unsigned int PORT\_MODBUS\_TCP = 502;**

***const* unsigned int LG\_MBAP = 7;**

***const* unsigned int FONCTION\_ECRIRE1BIT = 5;**

***const* unsigned int LG\_DEMANDE\_ECRIRE1BIT = LG\_MBAP+5;**

***const* unsigned int LG\_REPONSE\_ECRIRE1BIT = LG\_MBAP+5;**

***const* unsigned char FONCTION\_LECTURENMOTS = 3;**

***const* unsigned int LG\_DEMANDE\_LECTURENMOTS = LG\_MBAP+5;**

***const* unsigned int LG\_REPONSE\_LECTURENMOTS = LG\_MBAP+2;**

***const* unsigned char FONCTION\_ECRIRENMOTS = 16;**

***const* unsigned int LG\_DEMANDE\_ECRIRENMOTS = LG\_MBAP+6;**

***const* unsigned int LG\_REPONSE\_ECRIRENMOTS = LG\_MBAP+5;**

***const* unsigned int REPONSE\_OK=0;**

***const* unsigned int TIMEOUT=2000;**

***const* unsigned int ERREUR\_TIMEOUT=-257;**

***const* unsigned int LG\_ERREUR=LG\_MBAP+2;**

**class EsclaveModbusTCP {**

**private:**

**QString \_adresseIP;**

**int \_numEsclave;**

**QTcpSocket \_socket;**

**unsigned short \_transactionId;**

**unsigned char\* \_trameDemande;**

**unsigned char\* \_trameReponse;**

**unsigned char \_fonctionModbus;**

**void connecter();**

**void afficher (unsigned char*\** trame, int nbOctets);**

**void remplirMBAP (int lgTrame);**

**int envoyerTrameDemande (int nbOctets);**

**int recevoirTrameReponse(int nbOctets);**

**public:**

**EsclaveModbusTCP(QString adresseIP, int numEsclave=0);**

**int ecrire1Bit (unsigned short adresse, int valeur);**

**int lireNMots (unsigned short adresse, unsigned int nbMots, unsigned short*\** tabMots);**

**int ecrireNMots (unsigned short adresse, unsigned int nbMots, unsigned short*\** tabMots);**

**};**

**#endif *// ESCLAVEMODBUSTCP\_H***

1. Didactique : Qui vise à instruire [↑](#footnote-ref-0)
2. - - - : Mes tâches à effectuer [↑](#footnote-ref-1)
3. IDE : Environnement de développement [↑](#footnote-ref-2)