Hugo TABARY, Edouard HAUTEMANIÈRE et Alexandre LEPRÊTRE

Pilotage d'un carillon 4 cloches via TCP/IP et Modbus/TCP

Sommaire :

But du TP :

Le but de ce TP est de faire sonner les 4 cloches mises à notre disposition. Celles-ci seront pilotées depuis une application dédiée. Il est à noter que nous avons eu des problèmes d'adressage IP dans notre projet, nous avons tenté de la réparer par une VM WinXP dédiée mais en vain. Nous avons donc dû utiliser une carte d'acquisition branchée directement sur les relais électriques du circuit. Cette mésaventure nous à fait perdre beaucoup de temps, nous avons mis une journée pour pouvoir trouver une alternative fonctionnelle pour notre projet.

Principe du TP :

Grâce à la documentation, nous avons constitué une trame qui nous permettra de tester la bonne mise en forme de notre trame grâce à une mise en forme convenable nous avons effectué des test dans un premier temps en la diffusant via une application QT C++ que nous devions faire. Nous avons finalement dû utiliser l'application C++ déjà produite pour pouvoir faire nos tests, nous avons ensuite commencé le développement de l'application QT C++ pour pouvoir terminer notre sujet.

Questions préliminaires :

1) Recherchez dans les documentations le format du protocole Modbus/TCP. L'expliquer.

   Le protocole Modbus / TCP **utilise le standard Ethernet 10 Mbps** pour véhiculer toute la structure des messages Modbus. Il **fonctionne** sur le même principe que le **Modbus** série (rs-485, RS-232, RS-422), mais la différence se situe au niveau de l'architecture. En effet, le modèle maître-esclave est remplacé par le modèle client-serveur et utilise le réseau **Ethernet**.

2) La carte ETZ 510 utilise des trames TCP/IP dans lesquelles est encapsulé le protocole Modbus/TCP. Présentez le principe du client / serveur TCP/IP. ETZ 510 est-elle un serveur ou un client au sens de TCP/IP ?

   La carte ETZ 510 fonctionne en tant que **serveur** dans le contexte du protocole Modbus/TCP, où il est constamment à **l'écoute** de connexions entrantes, **traitant** les requêtes des clients et fournissant les services associés.

3) Donnez la trame Modbus/TCP qui permet d'activer la cloche 1 (la plus grosse). Voir documentation dans l'armoire électrique de l'installation.

   0x00 0x00 0x00 0x00 0x06 0x00 0x06 **0x00 0x01** 0x00 0x12 **0x01** 0x01
   output—--> crc->

4) Quel est le port par défaut spécifié dans la norme du protocole Modbus / TCP ?

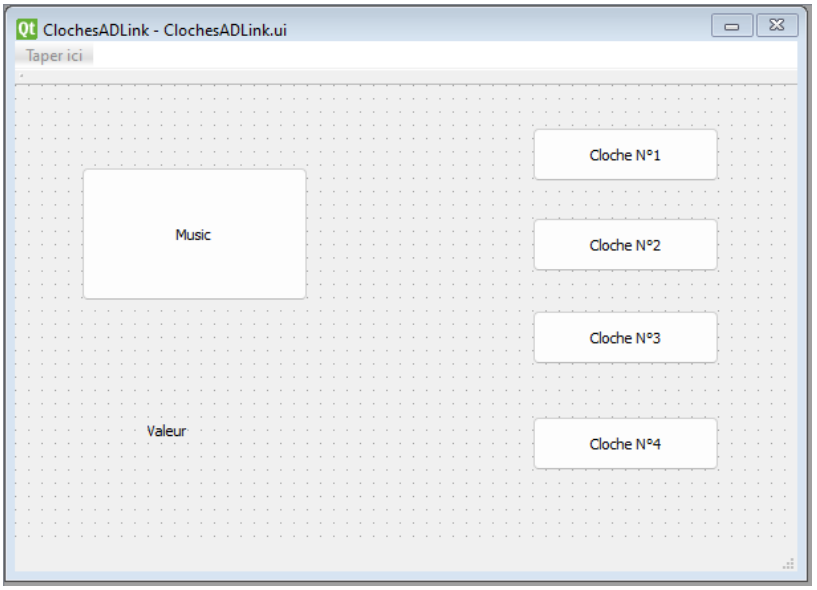
   Le port par défaut spécifié dans la **norme** du protocole Modbus / TCP est le **port 502**. C'est ce port qui est **généralement utilisé** pour les connexions avec un serveur Modbus / TCP.

5) Effectuer la connexion entre une application C++ et la carte ETZ via les objets client / serveurs de la librairie Qt. Quelle(s) classe(s) de Qt utilisez-vous pour réaliser cela ?

   Pour établir une **connexion** entre une application C++ et la carte ETZ via les objets client/serveur de la librairie Qt, on peut utiliser la classe **QTcpSocket** pour le client et **QTcpServer** pour le serveur.

Programme en C++ :

Impression des IHM :



L'IHM **(interface homme-machine)** a été conçue de façon simple et claire pour une utilisation rapide et intuitive.

Nous pouvons apercevoir 5 boutons, chacun ayant une fonctionnalité différente.

Les cloches N°1 à 4, notées de 0 à 3 dans le code, consiste à activer les cloches avec 2 positions. La première, "0", consiste à actionner le bras pour faire sonner la cloche, et le "1" pour réarmer le bras avec un délai de 500 ms.
La fonction Music permet d'activer les cloches dans un ordre prédéfini afin de créer une musique.

Impressions d'écrans types :

```
//*****************************************************************************************
//* Programme : ClochesADLink.cpp                          Date : 24/11/2023
//*-----------------------------------------------------------------------------------------
//* Dernière mise à jour : 24/11/2023
//*
//* Programmeurs : Alexandre Lepretre                      Classe : BTSSN2
//*                Hugo Tabary
//*                Edouard Hautemanière
//*-----------------------------------------------------------------------------------------
//* But : C'est l'endroit où sont stockées les méthodes de la classe ClochesADLink
//* Programmes associés : ClochesADLink.h
//*****************************************************************************************
```

Voici les commentaires d'entete inscrits en début de chaques fichiers du projet.

```
class ClochesADLink : public QMainWindow
{
    Q_OBJECT

public:
    ClochesADLink(QWidget *parent = nullptr);
    ~ClochesADLink();

private:
    Ui::ClochesADLinkClass ui;
    I16 cardId;

protected:
    virtual void keyPressEvent(QKeyEvent* ev);

public slots:
    void LauncherBellNum1();
    void LauncherBellNum2();
    void LauncherBellNum3();
    void LauncherBellNum4();
    void LauncherBellAll();
    void Label1();
};
```

Définition de la classe C++ ClochesADLink

```
#include "ClochesADLink.h"

ClochesADLink::ClochesADLink(QWidget *parent)
    : QMainWindow(parent)
{
    ui.setupUi(this);

    cardId = Register_Card(PCI_7256, 0);
}

ClochesADLink::~ClochesADLink()
{
    Release_Card(cardId);
}
```

Ici nous déclarons le type de la carte d'acquisition que nous utilisons. Celle-ci est défini dans le fichier Dask.h

```
void ClochesADLink::LauncherBellNum1()
{

    DO_WriteLine(cardId, PCI_7256, 0, 1);
    Sleep(500);
    DO_WriteLine(cardId, PCI_7256, 0, 0);
    Sleep(500);

}
```

Fonction d'allumage de la cloche n°1.
Ici DO_WriteLine() prends 4 paramètres, à savoir : le cardId, le type de carte, la cloche (donc le bornier) et l'état (1 pour activation et 0 pour le réarmement)

Conclusion :

Lors de ce TP un certain nombre de problèmes étaient à déplorer dû à la vétusté du projet, à commencer par une erreur dans les informations fournies notamment au niveau de L'IP suite à la venue des nouveaux 1er année de CIEL, l'IP fixe de notre TP a été réattribuer ce problème a était réglé en allumant l'automate de notre TP et en redémarrant la VM concerné cela nous a permis de récupérer notre IP fixe.

Suite à cela nous avons donc essayé de nous connecter à l'automate avec réussite pour tester d'envoyer nos trame aprés plusieur essaie avec une trame théoriquement valide nous n'entendons toujours pas nos chère cloche en allant observer l'automate plus précisément nous avons observé que le voyant ERREUR clignote, après une conversation avec notre professeur qui a essayé d'opérer une maintenance depuis une VM Windows XP dédié à ce projet précis mais malheureusement cette VM n'était plus en état de fonctionner donc la décision a était prise de changer la méthode utilisée et de passer d'une méthode ou l'on passe par le réseau pour une méthode passant directement par la carte d'acquisition disponible sur le TP cela change principalement la méthode de communication à la place de trame nous appelons simplement une fonction.

ATTENTION: Dans le code d'adaptation, le Débug mode renvoyé le code -13 si le mauvais modèle de carte est inséré dans le code ou si la carte d'acquisition est fichu.

Mis à part ses problèmes l'ensemble du code est entièrement fonctionnelle on peut actionner les cloches à l'aide des boutons de L'IHM , du clavier (touche A Z E R T P).