

TP Système : Les Cloches

1. Présentation :

Nous avons pour projet de concevoir un programme qui puisse communiquer avec une carte réseau ETZ-510. Le but était d'envoyer une trame à cette automate pour faire sonner les différentes cloches.

2. Question préliminaires :

- 1) Pour communiquer avec l'automate, nous devons lui envoyer une trame modbus encapsulée dans une trame TCP/IP. Le protocole modbus se trouve dans la partie application du modèle OSI.

Le protocole modbus peut être encapsulé par plusieurs supports ,RS-232 , RS-485, RS-422 et Ethernet.

Fonctionnement du modbus:

Le maître parle à un esclave et attend une réponse ou le maître parle à l'ensemble des esclaves, sans attendre de réponse. Il ne peut y avoir sur la ligne qu'un seul équipement en train d'émettre.

Nous avons utilisé un mode half-duplex c'est-à-dire qu'il n'y a qu'une seule communication à la fois.

La requête contient :

- L'adresse de l'esclave à interroger
- Le code fonction qui indique le type d'action à exécuter
- La plage de bits
- Les données à envoyer

La réponse contient :

- L'adresse de l'esclave qui répond
- Un code fonction qui indique l'action effectuée
- Le nombre d'octets de données
- Les données lues

- 2) Le principe du protocole TCP/IP client serveur c'est une communication entre deux machines, le client et le serveur.

Le client initie l'échange et envoie une requête.

Le serveur est en permanence en écoute sur son port, dès qu'il détecte une requête, il la traite et répond au client.

Un serveur doit être capable de répondre à plusieurs clients en même temps.

La carte ETZ-510 est donc un serveur car elle ne fait que répondre à nos requêtes.

3) Trame qui permet de faire sonner la cloche 1 :

```
char trame[12];
trame[0] = 0x00;
trame[1] = 0x01;
trame[2] = 0x00;
trame[3] = 0x00;
trame[4] = 0x00;
trame[5] = 0x06;
trame[6] = 0x11;
trame[7] = 0x06;
trame[8] = 0x00;
trame[9] = 0x02;
trame[10] = 0x00;
trame[11] = 0x01;

QByteArray data(trame, 12);
socket->write(data);
```

4)

```
QtWidgetsApplication1::QtWidgetsApplication1(QWidget *parent)
: QMainWindow(parent)
{
    socket = new QTcpSocket(this);

    QObject::connect(socket, SIGNAL(connected()), this, SLOT(onSocketConnected()));
    QObject::connect(socket, SIGNAL(disconnected()), this, SLOT(onSocketDisconnected()));
    QObject::connect(socket, SIGNAL(readyRead()), this, SLOT(onSocketReadyRead()));

    // 192.168.65.103
    socket->connectToHost("192.168.64.124", 502);

    ui.setupUi(this);
}

void QtWidgetsApplication1::onSocketConnected() {

    ui.label->setText("Connecter");

};
```

Application demandée

2) On créer une connexion avec QTcpSocket dans le .h

```
class QtWidgetsApplication1 : public QMainWindow
{
    Q_OBJECT

public:
    QtWidgetsApplication1(QWidget *parent = Q_NULLPTR);
    QTcpSocket * socket;

private:
    Ui::QtWidgetsApplication1Class ui;

public slots:
    void onSocketConnected();
    void onSocketDisconnected();
```

```
QtWidgetsApplication1::QtWidgetsApplication1(QWidget *parent)
    : QMainWindow(parent)
{
    socket = new QTcpSocket(this);

    QObject::connect(socket, SIGNAL(connected()), this, SLOT(onSocketConnected()));
    QObject::connect(socket, SIGNAL(disconnected()), this, SLOT(onSocketDisconnected()));
    QObject::connect(socket, SIGNAL(readyRead()), this, SLOT(onSocketReadyRead()));

    // 192.168.65.103
    socket->connectToHost("192.168.64.124", 502);

    ui.setupUi(this);
}

void QtWidgetsApplication1::onSocketConnected() {

    ui.label->setText("Connecter");
```

Ensuite on connecte le socket avec l'adresse ip et le bon port.

3)

```
char trame[12];
trame[0] = 0x00;
trame[1] = 0x01;
trame[2] = 0x00;
trame[3] = 0x00;
trame[4] = 0x00;
trame[5] = 0x06;
trame[6] = 0x11;
trame[7] = 0x06;
trame[8] = 0x00;
trame[9] = 0x02;
trame[10] = 0x00;
trame[11] = 0x01;

QByteArray data(trame, 12);
socket->write(data);

char trame2[12];
trame2[0] = 0x00;
trame2[1] = 0x01;
trame2[2] = 0x00;
trame2[3] = 0x00;
trame2[4] = 0x00;
trame2[5] = 0x06;
trame2[6] = 0x11;
trame2[7] = 0x06;
trame2[8] = 0x00;
trame2[9] = 0x02;
trame2[10] = 0x00;
trame2[11] = 0x00;

QByteArray data2(trame2, 12);
socket->write(data2);
```

4) Pour activer les cloches avec les touches on utilise QKeyEvent:

```
void QtWidgetsApplication2::keyPressEvent(QKeyEvent *ev)
{
    if (ev->key() == Qt::Key_Z)
    {
        ui.label->setText("You Pressed Key " +ev->text());
    }
}
```