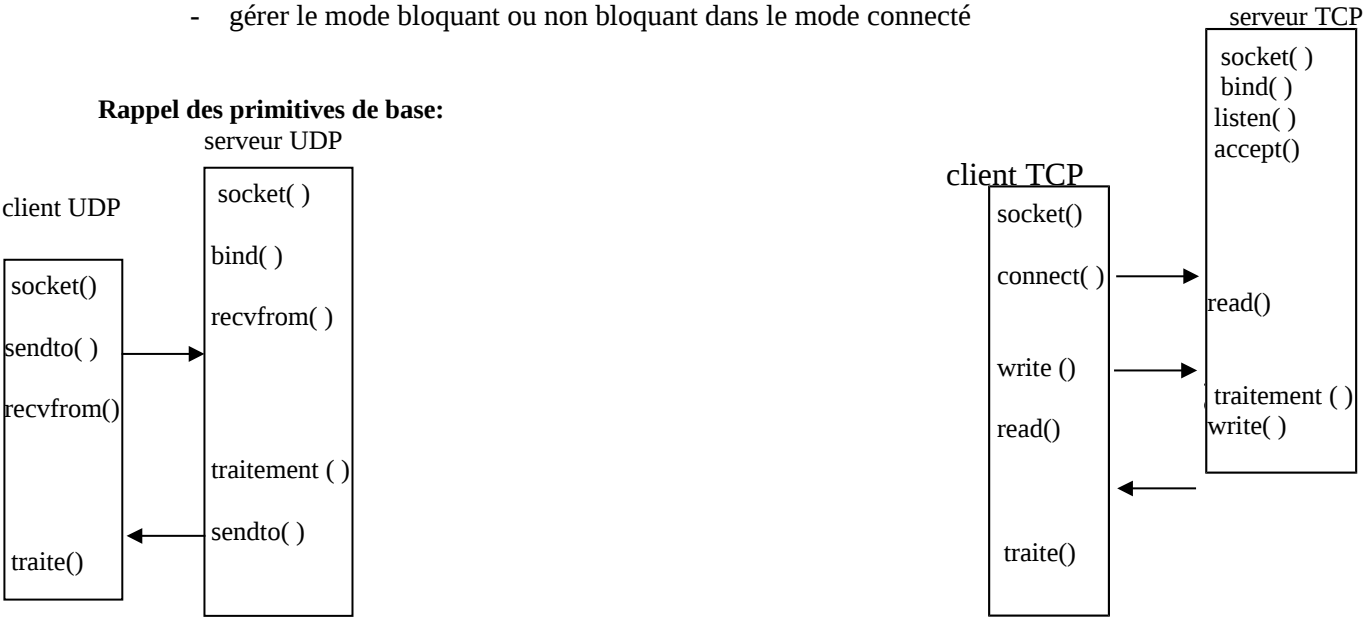
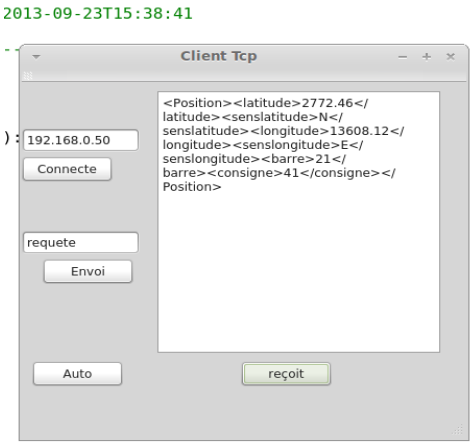
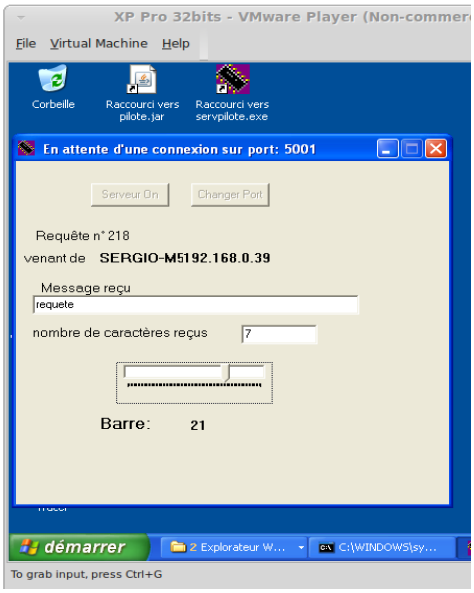
TP Socket **Communication Client/Serveur** Durée :6h

Objectifs :
- utiliser QT pour réaliser une communication client/serveur
- différencier le mode non connecté du mode connecté en **Tcp/Ip**
- gérer le mode bloquant ou non bloquant dans le mode connecté

Rappel des primitives de base:







But : Réaliser une application client qui envoie une requête à un serveur sur le port 5001 et affiche la réponse du serveur.
Dans le mode connecté la connexion sera ouverte avant chaque requête et donc fermée après chaque réponse.

Note : Afin de pouvoir utiliser la librairie " `QtNetwork` " vous devez ajouter QT += network dans votre fichier projet.pro:

```
QT       += core gui
QT += network
```

Mise en situation:
Vous disposez d'un serveur en écoute sur le port 5001 et de l'entête de la classe clientTcp en annexe 1.

Dans un premier temps, l''interface doit permettre de
- avec le bouton Connecte:
    instancier un `clientTcp` et appeler la méthode `connecte`
- avec le bouton Envoi et un LineEdit:
    saisir un message pour le transmettre à la méthode `envoi`

- avec le bouton reçoit:
appeler la méthode recoit et d'afficher le message retourné par le serveur dans un textEdit.
Suite à la réception, on appellera la méthode `deconnecte` et on détruira l'objet `client`

Travail demandé:

1) Créer un projet, ajouter l'accès à la librairie `QtNetwork,` insérer les composants visuels nécessaires.

2) Implémenter et tester les méthodes de la classe `ClienTcp en` ayant associé les événements et les appels des méthodes de la classe `clientTcp`

3) Modifications:
Ajouter et utiliser un nouveau constructeur qui mémorise l'adresse Ip et le port du serveur dans les attributs adéquats.
Ajouter et utiliser un méthode `connecte` ne prenant pas de paramètre. La connexion sera réalisée avec les valeurs des attributs.

4)Lecture des réponses serveur avec un timer (annexe 2)
- Ajouter un bouton " Auto " démarrant chaque seconde une communication avec le serveur.
La méthode `maj`(mise à jour) est appelée chaque fois que le timer a fini de compter.

4,1)Méthode bloquante
- Implémenter la méthode void MainWindow::maj

Vous devrez modifier la méthode recoit afin d'attendre que le buffer de réception soit plein avec la méthode
`QTcpSocket::waitForReadyRead`

4,2)Méthode non bloquante
La méthode `recoit` n'est plus bloquante: on n'utilise plus `waitForReadyRead`
Par contre le fait que le buffer de réception soit plein déclenche un événement.
Vous devez ajouter dans la méthode clic du bouton Auto

```
QTcpSocket *maSocket=client->getSocket();
connect(maSocket, SIGNAL(readyRead()), this, SLOT(lit()));
```

- Modifier la méthode `void MainWindow::maj` afin qu'elle ne fasse plus que la connexion et l'envoi du message.
- Implémenter la méthode `void MainWindow::lit` afin qu'elle affiche le message retourné par recoit et se déconnecte.

5) Réaliser un client UDP

Critères d'évaluation :
- Degré d'autonomie dans la recherche d'informations
- Maîtrise de l'outil de développement.
- Méthodes de développement : écriture d'une fonction et test...
- Avancement du travail
- Qualité du programme : commentaires synthétiques utiles, entête...

Annexe 1 Header de la classe clientTcp

```
#define CLIENTTCP_H
#include <QtNetwork/QTcpSocket>
#include <QHostAddress>

class clientTcp
{
public:
    clientTcp();// instancie un QTcpSocket pointé par monSocket
       //les autres attributs sont initialisés avec des valeurs par défaut 5001 pour le port et
       //" "  pour adIp

    void connecte(QString adServeur, qint16 port); //utilise la méthode QtcpSocket::connectToHost

       // afin d'établir la connexion

    void envoi(QString message); //utilise la méthode  QTcpSocket::write pour envoyer message

    QString recoit ();    // utilise la méthode  QTcpSocket::readAll pour lire le message reçu

    void Deconnecte();    // utilise la méthode  QtcpSocket::disconnectFromHost pour se
                   //déconnecter

    QTcpSocket * getSocket(); // retourne monSocket

 private:
    QTcpSocket * monSocket;
    qint16 port;
    QString adIp;
  };
#endif // CLIENTTCP_H
```

Annexe 2

```
       void MainWindow::on_Auto_clicked()
{
       client=new clientTcp();
       timer = new QTimer(this);
        connect(timer, SIGNAL(timeout()), this, SLOT(maj()));
       timer->start(1000);

}
```