Projet de mécatronique AMER2
Informations spécifiques

Nicolas Stouls **modifié le 05/05/10**

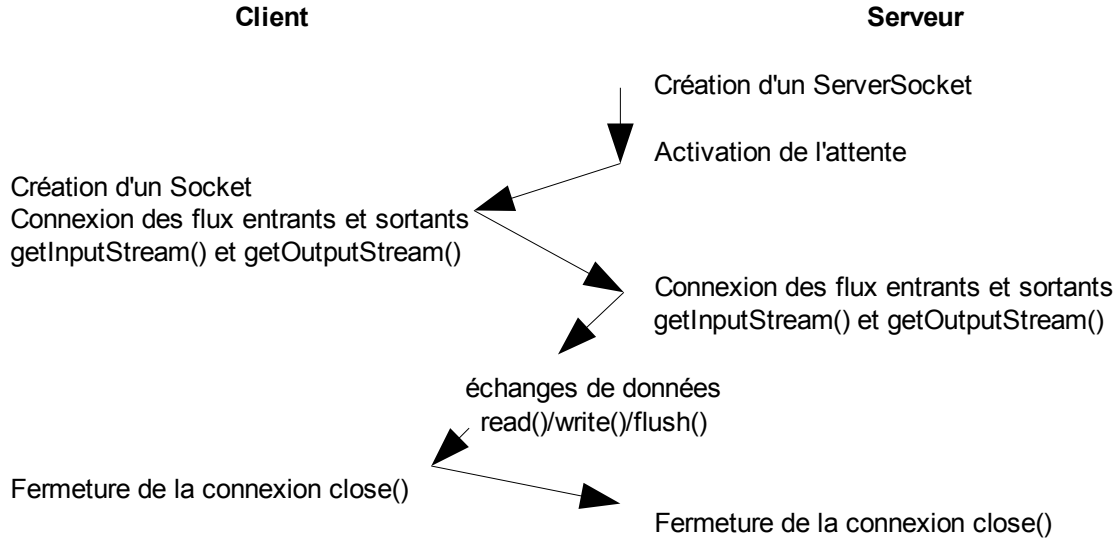
Table des matières

Dans ce document complémentaire spécifique au projet de mécatronique, nous vous donnons quelques informations supplémentaires. Notez que pour ce projet, les rendus attendus et les modalités de notation sont identiques à ceux des projets « traditionnels » réalisés dans les autres groupes. La différence principale est que vous devrez également fournir un micro-code pour la programmation du micro-contrôleur. Ce code sera réalisé par un groupe transversal : composé d'un représentant de chaque groupe.

De plus, le prototype réalisé dans le cadre du projet de mécatronique est dirigé par un micro-contrôleur qui a la tâche de traiter les informations capturées par les capteurs pour les transmettre au PC, ainsi que de recevoir les ordre de l'ordinateur et de les transmettre à ses périphériques actifs. Vous serez donc amenés à réaliser un module de communication réseau. La connexion avec ce micro-contrôleur se fait via une connexion réseau de Type TCP/IP.

1. Bases de TCP/IP

Nous ne rentrerons pas dans les concepts fins des connexions TCP/IP, mais nous voulons pouvoir écrire le code d'un **client**, car le micro contrôleur sera configuré comme étant un serveur. Nous devons donc comprendre le déroulement général d'une communication :



- Créer le java.net.Socket :
  - Socket server = new Socket(host,port);
- Récupèrer les flux d'entrée et de sortie :
  - InputStream in = server.getInputStream();
  - OutputStream out = server.getOutputStream();
- Échanger en mode Bloc de bytes :
  - Classes java.io.DataInputStream/DataOutputStream

Exemple de code d'un client tiré du cours de Didier Donsez :

```
import java.net.*; import java.io.*;
public class EchoClient {
  public static void main(String[] args) {
    DataInputStream tcpIn;
    try {
      Socket theSoc = new Socket(args[0], Integer.parseInt(args[1]));
      tcpIn = new DataInputStream(theSoc.getInputStream());
      PrintStream tcpOut = new PrintStream(theSoc.getOutputStream());
      DataInputStream userInput = new DataInputStream(System.in);
      while (true) {
        String theLine = userInput.readLine();
        if (theLine.equals(".")) break;
        tcpOut.println(theLine);
        System.out.println(tcpIn.readLine());
      }
    } catch (UnknownHostException e) { System.err.println(e); }
      catch (IOException e) { System.err.println(e); } } }
```

Afin de tester votre couche réseau, nous mettons à votre disposition une Applet configurée en serveur TCP, permettant de simuler la communication entre votre programme et le micro contrôleur.

2. Adresse des micro-contrôleurs

Le micro contrôleur utilisé en IF a l'adresse suivante :

```
adresse IP : 134.214.152.178
Nom IP : cc107-01.insa-lyon.fr.
Masque : 255.255.252.0
Passerelle : 134.214.152.1
```

L'adresse du micro-contrôleur utilisé en GCP n'est pas encore connue.

3. Protocole d'échange proposé

Tous les messages échangés entre l'ordinateur et le micro-contrôleur sont composés d'un code opération sur 8bits et d'un paramètre (entier sur 16 bits).

Attention également aux spécificités de Java. En effet, les entiers utilisés dans notre protocole sont toujours non signés, tandis que les entiers java sur 1 ou 2 octets sont toujours signés. Ce sera à vous de faire la conversion. Les domaines de définition et encodages des données utilisées sont décrites en fin de section.

Description du protocole ci-dessous :

```
La première ligne est l'envoie ordinateur vers micro-contrôleur
La ligne en italique est le retour micro-contrôleur vers ordinateur
```

Commandes micro-contrôleur

codeOp	Param	Description
01	*pos.*	rotation tourelle à la position *pos.*
01	*pos.*	*Commande effectuée (La position réelle est inconnue).*
02	0000	ouverture vanne remplissage
02	*0001*	*Commande effectuée (La valeur réelle est inconnue).*
03	*pres.*	fermeture vanne remplissage si pression > *pres.*
03	0000	fermeture immédiate vanne remplissage
03	*pres*	*La valeur de retour est la pression atteinte*
04	*tps.*	délais d'attente
04	*tps.*	*délais d'attente (pas de contrôle)*
05	0000	demande de renvoie de la valeur de pression
05	*pres*	*renvoie de la valeur de pression*
06	0000	ouverture vanne de tir
06	*0001*	*vanne de tir commandée (pas de contrôle possible)*
07	0000	fermeture vanne de tir
07	*0001*	*vanne de tir commandée (pas de contrôle possible)*
		Commandes actives avec suivi de progression
0A	*pres*	fermeture vanne remplissage si pression > *pres*
0A	*pres*	*Acquittement commande*
0D	*tps*	*+ renvoi valeur temps en continu 1 message sur 6*
0C	*pres*	*+ renvoi la pression en continu tous les 1/1200 s Répétition 0D/0C tous le 6ms environ jusqu'à consigne de pression atteinte.*
0A	*FFFF*	*En fin de cycle*
0B	*tps*	délais d'attente
0B	*tps*	*Acquittement commande*
0D	*tps*	*+ renvoi valeur temps en continu 1 message sur 6*
0C	*pres*	*+ pression en continu tous les 1/1200 s Répétition 0D/0C tous le 6ms environ jusqu'à consigne de temps atteinte.*
0B	*FFFF*	*En fin de cycle*

Codes de gestion de la communication

codeOp	Param	Description
Les messages d'erreur (7D) sont suivi d'une copie du message qui a entrainé l'erreur		
7D	*FFFF*	*longueur de trame non multiple de 3*
7D	*FFFE*	*code inexistant*
7D	*FFFD*	*timeout pression de consigne non atteinte*
7D	*FFFC*	*paquet de commandes trop long*
7C	0000	demande de maintien de la communication
7C	*0001*	*acceptation de maintien de la communication*
80	00FF	fin de communication (Question et réponse)

Codes de test		
codeOp	**Param**	**Description**
7E	*val.*	demande d'incrément de la valeur 8 bits *val.*
7E	*val.*	*réponse à la demande d'incrément entier*
7F	*val.*	demande d'incrément de la valeur 16bits *val.*
7F	*val.*	*réponse à la demande d'incrément entier*

Domaine de définition et encodage des données :

- Position tourelle : 128 est la position médiane, 0 correspond à –90° et 255 à +90°.
- Pression : codée sur 10 bits. La valeur maximale (1024) correspond à 7 bars.
- Temps : valeur en 1/1200 s (valeur transmise/1200 = temps en seconde)

Exemple de trames à envoyer pour effectuer un lancer (codes en hexadécimal) :

01 00 80 **02** 00 00 **03** 01 F0 **06** 00 00 **04** 02 00 **07** 00 00 **80** 00 FF

Trame renvoyée par le microcontrôleur après exécution

01 *00 80* ***02*** *00 01* ***03*** *01 F1* ***06*** *00 01* ***04*** *02 00* ***07*** *00 01* ***80*** *00 FF*

4. Programmation du micro-contrôleur

Le micro-contrôleur est programmé avec le logiciel Flowcode.

Un programme de base est fourni.

Ce programme reçoit, interprète et retourne correctement les trames reçues selon le protocole d'échange proposé. Les entrées et sorties vers le système mécanique ne sont pas programmées. Ce travail reste à faire.

Particularités du code proposé :

- Le port utilisé est le 5000.
- Le nombre d'octets maximum de la trame est de 100.
- Une seule connexion possible à la fois.
- La réponse est renvoyée une fois toutes les action effectuées.
- Si aucun échange de données n'est réalisé avec l'ordinateur, ce dernier est déconnecté au bout d'une minute.
- Si la pression de consigne n'est pas atteinte après 32000/1200=27s, un code d'erreur « 7D 65533 » est renvoyé.
- Au début de l'exécution de la trame, le temps renvoyé vaut 0. La base de temps est le 1/1200 s.
- Le programme est prévu pour le PIC 18F2680. La carte ethernet EB023 doit être branchée sur le port C. Le triple jumper de la carte EB023 doit être en position A.