Sommaire

1 - Objectifs

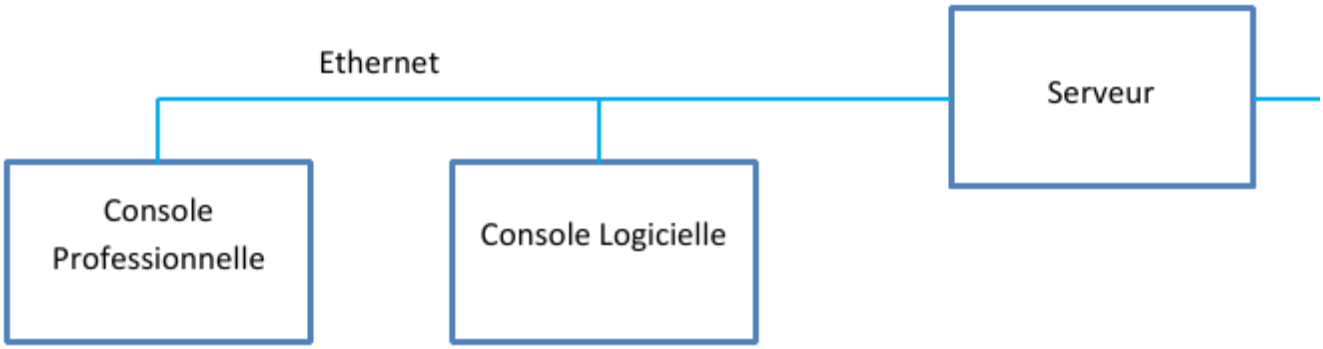
1.1 Présentation

Il est demandé de réaliser un programme (serveur) capable de communiquer avec un équipement professionnel (client). La contrainte de développement fixée pour ce TP est d'utiliser le support *socket* fourni par **Qt**. On ne prévoit pas d'IHM GUI pour ce serveur.

L'équipement professionnel choisi sera une **console Enttec**.

Une **console logicielle** (qui simule une console Enttec) est fournie afin d'assurer des tests de communication pour le programme serveur à réaliser (cf. Annexe).

1.2 Schéma



2 - Présentation de la console Enttec

La console Enttec *playback* est équipée d'une carte réseau et elle est capable de communiquer en **UDP/IP**.

2.1 Caractéristiques

ADRESSE IP : 10.7.89.95

PORT : 3330

MASQUE : 255.0.0.0

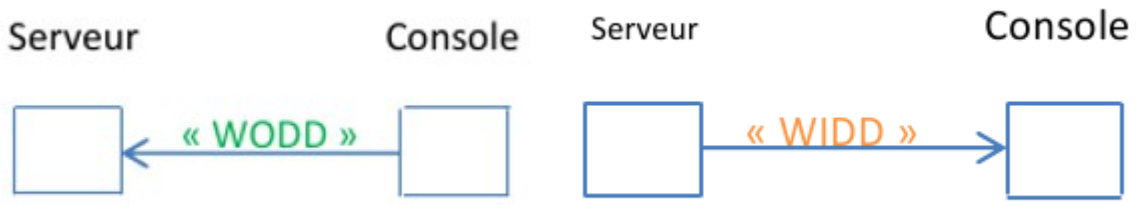
ADRESSE BROADCAST : 255.255.255.255

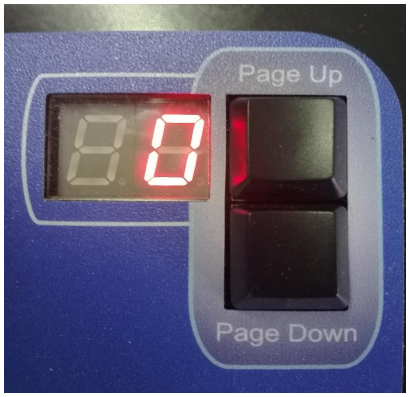
2.2 Protocoles

La console Enttec utilise un **protocole propriétaire WING** comme protocole de **couche Application**. Celui est décrit précisément dans le fichier **wing_api_spec.pdf**.

Il s'appuie sur **deux types de messages** :

- le message de type **WODD** est envoyé par la console à chaque appui sur une touche ou une modification des contrôle de type *slider*. Il contient l'état de l'ensemble des touches de la console.
- le message de type **WIDD** est reçu par la console pour commander les 2 afficheurs 7 segments. Il contient la valeur à afficher.





Le **transport** des messages WODD/WIDD est assuré par le protocole **UDP**. Le **modèle DoD** est donc le suivant :

Application	WODD/WIDD
Transport	UDP
Réseau	IP
Interface	Ethernet_II

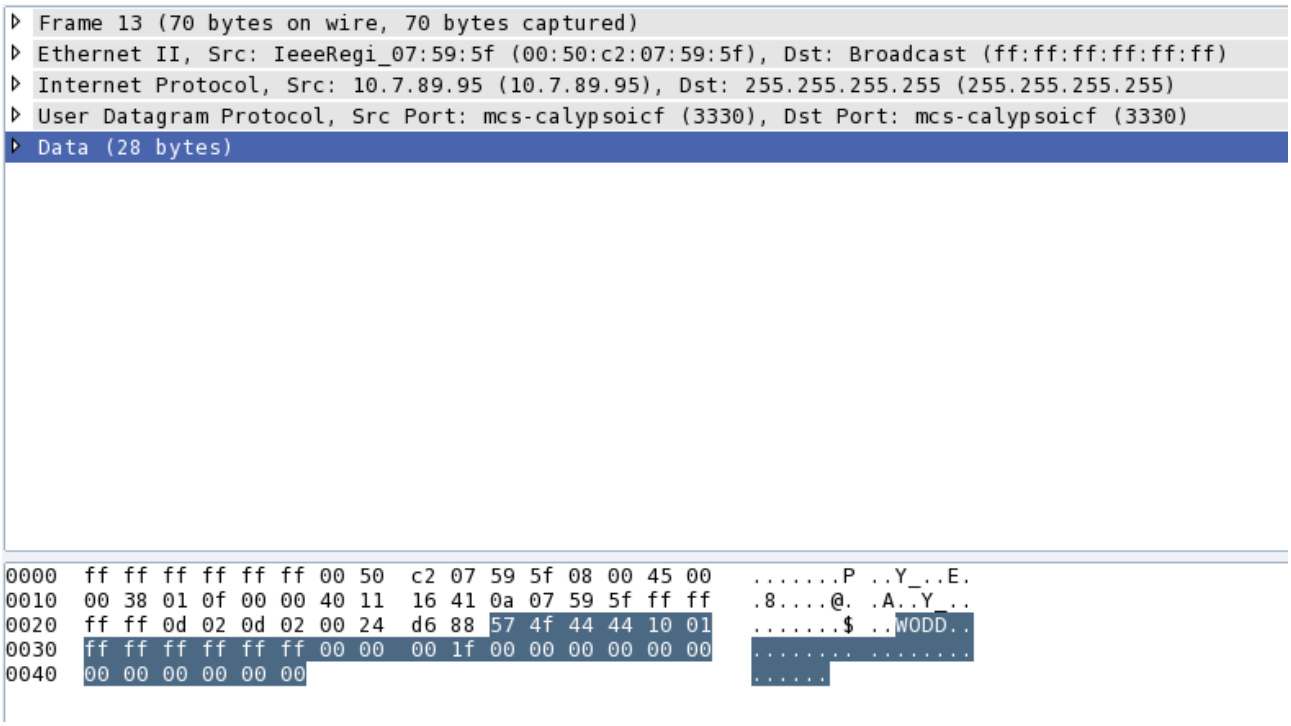
2.3 Capture de trame réseau

Voici un exemple de trame envoyée par la console Enttec lors de l'appui sur une touche :

```
ff ff ff ff ff ff 00 50  c2 07 59 5f 08 00 45 00
00 38 06 2f 00 00 40 11  11 21 0a 07 59 5f ff ff
ff ff 0d 02 0d 02 00 24  d7 27 57 4f 44 44 10 01
ff ff ff ff ff 7f 00 00  00 ff ff ff ff ff ff ff
ff ff ff 00 00 00
```

2. Décodage de trame réseau

Le décodage de cette trame par *wireshark* :

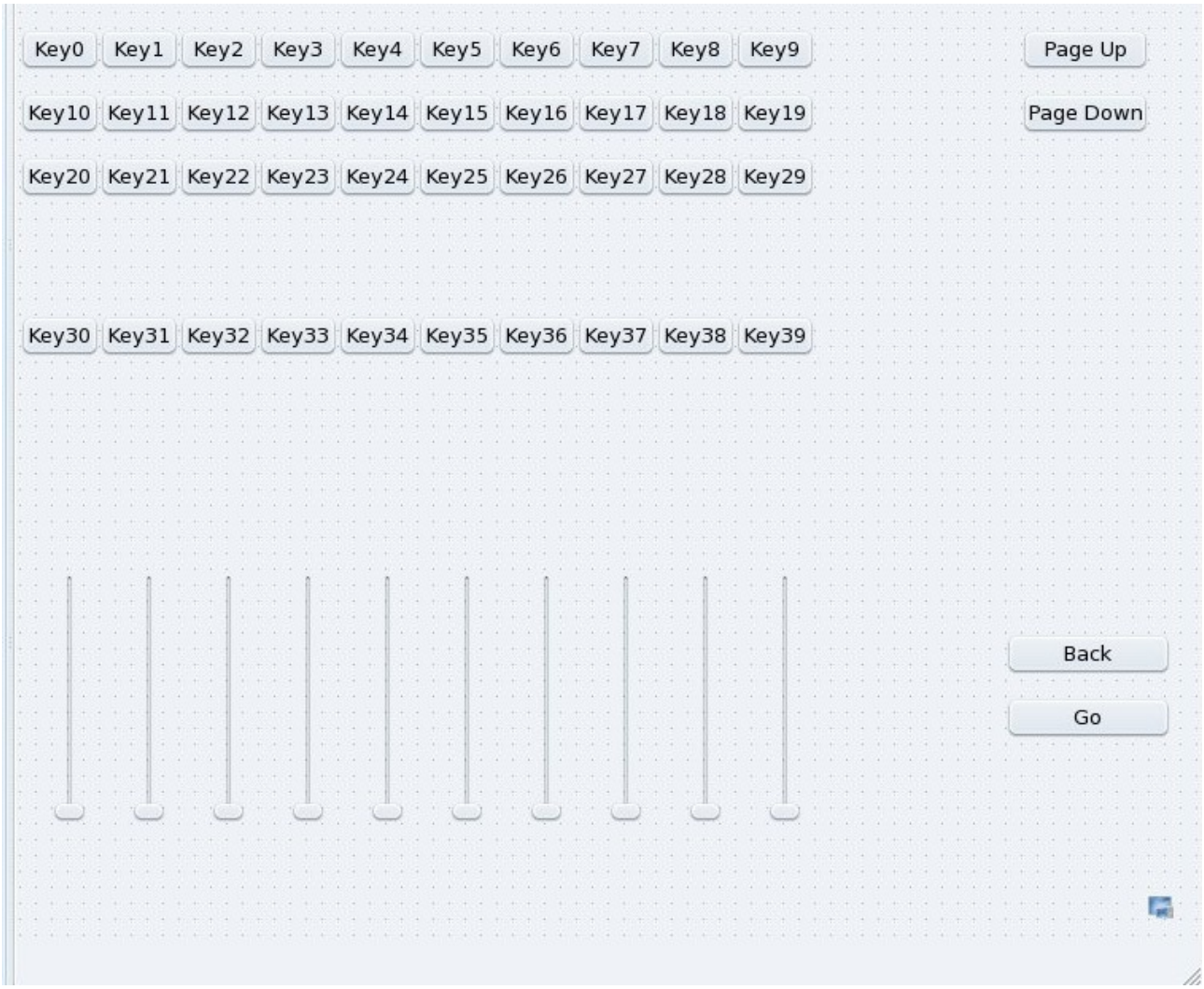


En résumé, la console Enttec émet ses datagrammes en ***broadcast* IP** (adresse destination 255.255.255.255) vers le port **UDP 3330**. Son port d'écoute pour la réception de datagramme est aussi le 3330.

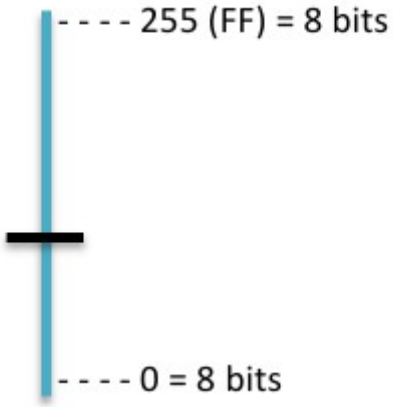
*Remarque : les 28 octets de données sont « décodables » à partir de la documentation Enttec dans le fichier **wing_api_spec.pdf**.*

2.3 Affectation des touches

Les touches de la consoles Enttec sont identifiables de la façon suivante :



Les curseurs (en bas sur la figure ci-dessus) sont appelés des *faders*. Ils sont numérotés de 0 à 9, de gauche à droite. Ils renvoient une valeur sur 8 bits suivant la position du curseur.



3 - Travail demandé

3.1 Sujet

Le programme serveur à réaliser en Qt/C++ doit être capable :

- de recevoir et de décoder (au moins afficher) des messages WODD
- d'envoyer des messages WIDD

On peut utiliser soit la console Enttec soit le programme simulateurEnttecUdp fourni.

On utilise **`QUdpSocket`** pour dialoguer avec la console.

La fabrication de l'application serveur se fera de la manière suivante :

```
$ qmake
$ make
```

Le code source de la fonction principale **`main()`** est fourni en Annexe.

Il vous faut respecter le fichier **`main.cpp`**, le diagramme de classe et le diagramme de séquence partiels fournis ci-après.

Documentation Qt pour `QUdpSocket` :

- http://developer.qt.nokia.com/doc/qt-4.7/qudpsocket.html

Quelques informations complémentaires sur la classe `DialogueConsole` à réaliser :
Le constructeur de la classe **`DialogueConsole`** doit assurer la création de la socket (udpSocket) et son attachement sur le **port 3330 des interfaces locales**.

La méthode **`demarrer()`** (appelée à partir de la fonction **`main()`**) réalise la connexion du ***signal*** readyRead() au ***slot*** ReceptionnerDatagrammes() (si l'attachement **`bind()`** a évidemment réussi).

La méthode **`VerifierDatagramme()`** permet de s'assurer de la validité du datagramme reçu. Pour cela, on vérifiera que le type du message est bien '**WODD**' et que sa taille correspond à celle fournie par le protocole WING. Cette méthode retourne 1 si le datagramme est valide sinon 0.

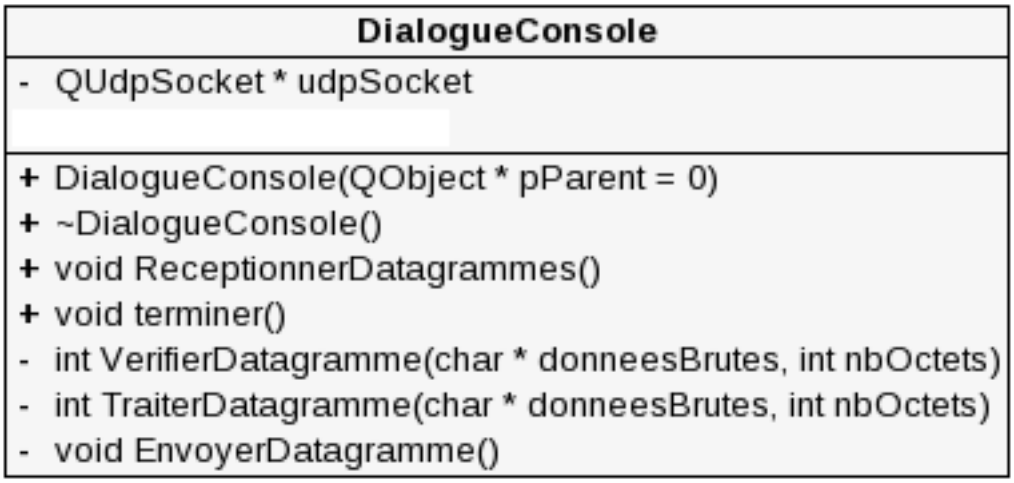
La méthode **`TraiterDatagramme()`** doit assurer l'affichage simple des données reçues (voir exemple fourni en Annexe) et de la détection des touches PAGE_UP ou PAGE_DOWN. De plus, elle retournera 1 si une touche PAGE_UP a été appuyée, 2 si c'est la touche PAGE_DOWN ou 0 si aucun appui sur ces deux touches.

La méthode **`EnvoyerDatagramme()`** permet l'envoi d'un message de type 'WIDD' vers la console. Elle est invoquée si une touche PAGE_UP ou PAGE_DOWN a été appuyée. Le message contiendra la valeur qui permettra d'incrémenter ou décrémenter l'affichage sur la console.

La méthode **`ReceptionnerDatagrammes()`** assure la réception UDP. Pour cela, elle réalise un boucle d'attente sur la présence de données dans la socket. Son pseudo code à compléter est le suivant :

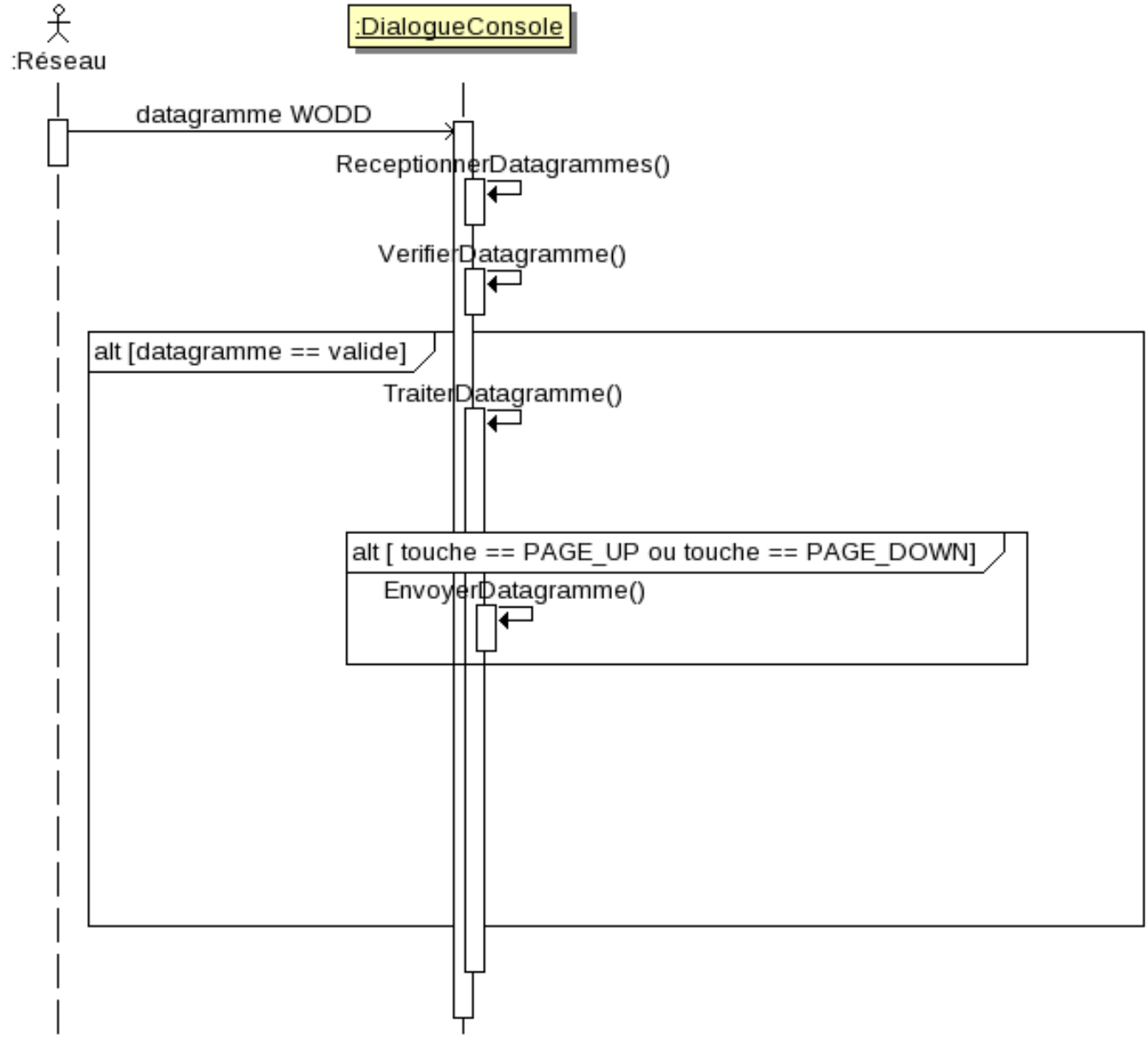
```
// Retourne vrai si au moins un datagramme est en attente d'être lu
   while (udpSocket->hasPendingDatagrams())
   {
      QByteArray donneesDatagramme;

      // Fixe la taille du tableau au nombre d'octets reçus en attente
      // cf. pendingDatagramSize()

      // Lit le datagramme en attente cf. readDatagram()

      // Vérifie la validité du datagramme
      etat = VerifierDatagramme(donneesDatagramme.data(), nbOctets);
      if(etat == 1)
      {
        touche = TraiterDatagramme(donneesDatagramme.data(),
                                           donneesDatagramme.size());
        if(touche > 0)
           // Envoie un datagramme avec un message WIDD
      }
      else
      {
        emit terminer(); /* sinon on quitte !  */
      }
   }
```

3.2 Diagramme de classe



Remarque : diagramme de classe partiel à compléter si nécessaire.

3.3 Diagramme de séquence



Remarque : diagramme de séquence partiel à compléter si nécessaire.

Annexes

Fichier projet Qt

Le fichier .**pro** sera le suivant :

```
TEMPLATE = app
TARGET = serveurUDP
DEPENDPATH += .
INCLUDEPATH += .
QT += network

HEADERS += DialogueConsole.h
SOURCES += main.cpp DialogueConsole.cpp
```

Fichier main.cpp

Le code source du fichier **main.cpp** est le suivant :

```
#include <QApplication>
#include "DialogueConsole.h"

int main(int argc, char *argv[]) {
   int ret = 0;
   DialogueConsole *dialogueConsole;
   QApplication app(argc, argv);

   dialogueConsole = new DialogueConsole();
   dialogueConsole->demarrer();

   ret = app.exec();
   delete dialogueConsole;
   return ret;
}
```

Exemple

Voici la sortie « écran » d'un programme serveur d'exemple :

```
<DialogueConsole.cpp:DialogueConsole:23> socket UDP sur le port 3330

<127.0.0.1:3331> datagramme de 28 octet(s) reçu(s)
WODD 0x10 0x01 0xFF 0xFF 0xFF 0xFF 0xFF 0xFD 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
<127.0.0.1:3331> datagramme de 28 octet(s) reçu(s)
WODD 0x10 0x01 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
<127.0.0.1:3331> datagramme de 1 octet(s) reçu(s) : INVALIDE !
```

Console logicielle

Deux programmes de test sont fournis pour simuler la console Enttec :

- **communicationEnttec.c**
  C'est un client UDP qui permet le dialogue avec la console Enttec en gérant les deux types de messages : WODD et WIDD. Pour quitter ce programme, il faut appuyer sur le touche BACK. Les touches PAGE_UP et PAGE_DOWN permettent la commande de l'afficheur 7 segments (valeur comprise entre 0 et 99). Sinon, le programme affiche les états des touches et *faders* reçus dans un message de type WODD.

  Compilation : `gcc communicationEnttec.c -o communicationEnttec`

- **simulateurEnttecUdp.c**
  Ce programme (non terminé) permet de simuler une console Enttec dans un mode très simplifié. Les touches KEY_0 à KEY_7 sont remplacés par les touches azertyui du clavier. La touche q permet de quitter le programme (en envoyant un datagramme UDP non valide).

  Compilation : `gcc simulateurEnttecUdp.c -o simulateurEnttecUdp`