MODULE 06

SÉANCE SYSTÈME 07

TP D'INFORMATIQUE

Durée 2h30

INTERFACE GRAPHIQUE : DÉTECTION AUTOMATIQUE DES CAPTEURS ULTRASON ET AFFICHAGE DES PARAMÈTRES

BLOC DE COMPÉTENCES

U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)

C08 - CODER

OBJECTIFS PÉDAGOGIQUES

Codage d'une interface graphique permettant d'interroger une plage de capteur afin de dresser la liste des appareils connectés. Supervision d'un capteur : lecture et écriture des paramètres.

CONNAISSANCES ISSUES DU RÉFÉRENTIEL

- Langages de développement, de description, de création d'API et les IDE associés — Niveau 3
- Programmation orientée objet — Niveau 3

CONNAISSANCES OPÉRATIONNALISÉES

- Concevoir une interface graphique sous Windows à partir de la documentation d'un matériel. — Niveau 2
- Versionner un code — Niveau 2

Gaëtan GEORGES
Pascal HAGOT
David IERVESE

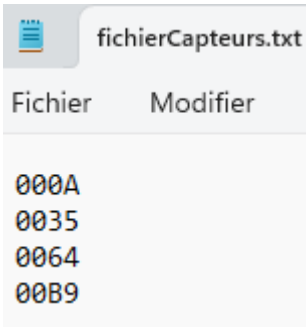
Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

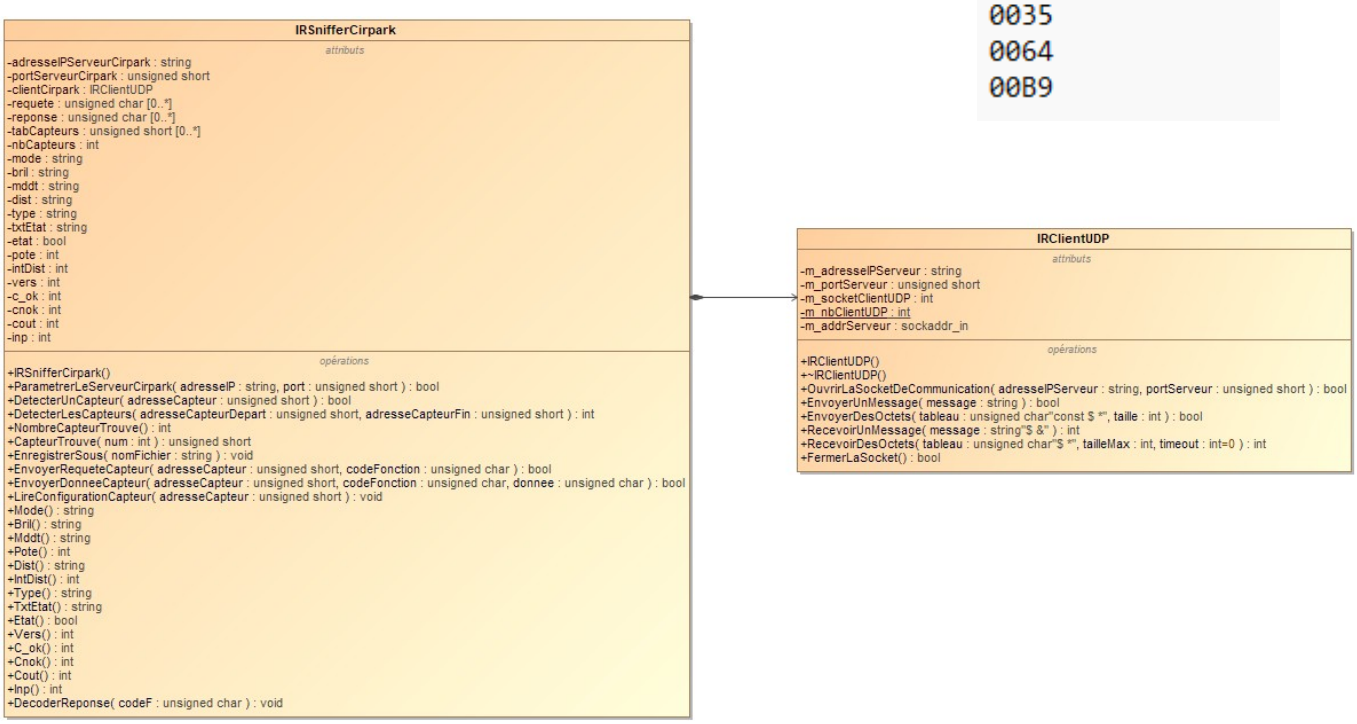
TP
Intégration dans une interface graphique de la classe IRSnifferCirpark : recherche des capteurs

Créer sous C++ Builder, une nouvelle application C++. Ajouter au projet les fichiers nécessaires au fonctionnement de la classe IRSnifferCirpark. Dans la fenêtre de conception, ajouter une zone Edit (qui contiendra l'IP du serveur UDP), un Label et un Button. Dans la classe TForm1 (unit1.h), ajouter, en attribut un objet sniffer de la classe IRSnifferCirpark. Générer l'événement Button1Click. Dans cet événement (unit1.cpp), appeler successivement les méthodes de la classe IRSnifferCirpark :

- ParametrerLeServeurCirpark : en paramètre AnsiString(Edit1->Text).c_str() et 10001 (le port)
- DetecterLesCapteurs : en paramètre la plage de recherche indiquée par l'enseignant
- NombreCapteurTrouve : dont le résultat sera placé dans le Caption du Label
- EnregistrerSous : en paramètre le nom du fichier texte au choix qui contiendra la liste des cateurs

Tester le fonctionnement en vérifiant la liste et le nombre de capteurs.



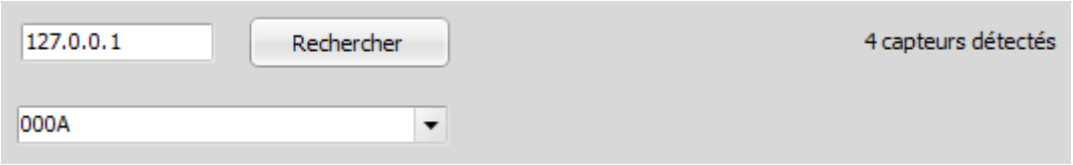


TP
Lecture de la configuration d'un capteur

Lecture du fichier texte contenant la liste des capteur, et placement des adresses dans un ComboBox

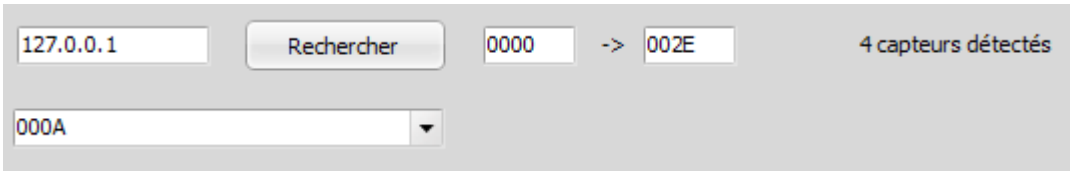
> Toujours dans l'événement Button1Click, Créer un objet de la classe ifstream afin de lire le fichier texte, appeler la méthode open avec cet objet, puis placer chaque adresse de capteur dans le ComboBox avant de refermer la fichier (méthode close()) : on donne :

```
          char adresseC[5];
          for(int i=0;i<sniffer.NombreCapteurTrouve();i++)
          {     f>>adresseC;
                ComboBox1->AddItem(adresseC,this);
          }
          ComboBox1->Text=adresseC;
```



Bonus : paramétrage de la plage de recherche

> Ajouter 2 zones Edit afin de définir la plage de recherche dans l'interface. Utiliser la méthode ChaineHexaVersInt() pour transmettre les 2 valeurs à la méthode DetecterLesCapteurs().

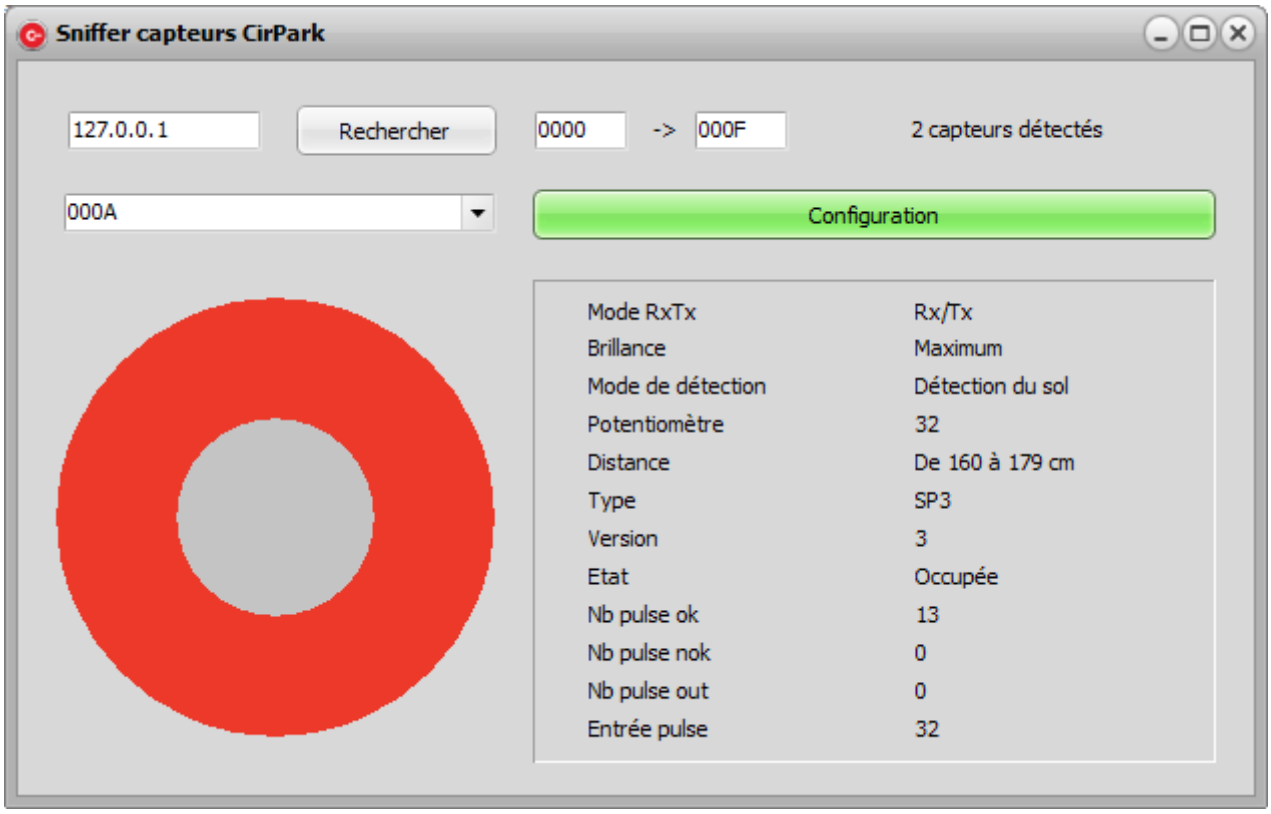


On rappelle la méthode ChaineHexaVersInt() (voir module Exxotest) qu'il est nécessaire d'ajouter à la classe de l'IHM afin de convertir les 4 caractères hexa au format ASCII en nombre :

```
unsigned int TForm1::ChaineHexaVersInt(char *chaineHexa, int nbCar)
{  int nombre=0,j;
   char caractere;
   for(j=0;j<nbCar;j++)
   {   caractere=chaineHexa[j];
       if(caractere>='0' && caractere<='9')
       {  nombre=nombre+((caractere-'0')<<(4*(nbCar-j-1)));
       }
       else nombre=nombre+((caractere-'A'+10)<<(4*(nbCar-j-1)));
   }
   return nombre;
}
```

Lecture et affichage graphique de la configuration d'un capteur

Placer dans la fenêtre de conception les éléments nécessaires à l'affichage graphique des paramètres d'un capteur.



Lors de l'appui sur le bouton Configuration, appeler la méthode LireConfigurationCapteur de l'objet sniffer, en lui passant en paramètre :

```
ChaineHexaVersInt(AnsiString(ComboBox1->Text).c_str(),4)
```

Affecter à chaque Caption des Labels les paramètres retournées par les méthodes de l'objet sniffer :

```
 LabelN->Caption=sniffer.Mode().c_str(); //N étant le numéro du label
...
```

L'anneau rouge est composé de 2 Shapes superposés. Selon l'état du capteur (sniffer.Etat()), modifier le paramètre Shape1->Brush->Color afin de le faire passer du rouge (occupé) au vert (libre).

TP
Codage d'un thread

Lors de la recherche d'un capteur, l'application ne répond plus au système d'exploitation. Il est nécessaire de placer la code de la recherche (les requêtes UDP dans un thread.

> Qu'est ce qu'un thread ?

> Créer la classe thread : Fichier->Nouveau->Autre...->Objet thread : classe ThRecherche. Une nouvelle unité (unit2) est créée avec son .h et son .cpp.

> Afin que les 2 classes communiquent entre elles, inclure unit2,h dans le fichier unit1,h et unit1,h dans unit2.cpp. Dans unit1.h, ajouter à TForm1 l'attribut ThRecherche *maThread.

> Placer dans la méthode Execute() de la classe thread le code du bouton Rechercher de l'interface graphique. Il faudra ajouter devant chaque élément de la classe TForm1 : Form1→. Exemple :

```
Form1->sniffer.ParametrerLeServeurCirpark(AnsiString(Form1->Edit1->Text).c_str(),10001);
```

> Tester le fonctionnement : l'application graphique ne se fige plus.

> Versionner le code complet.

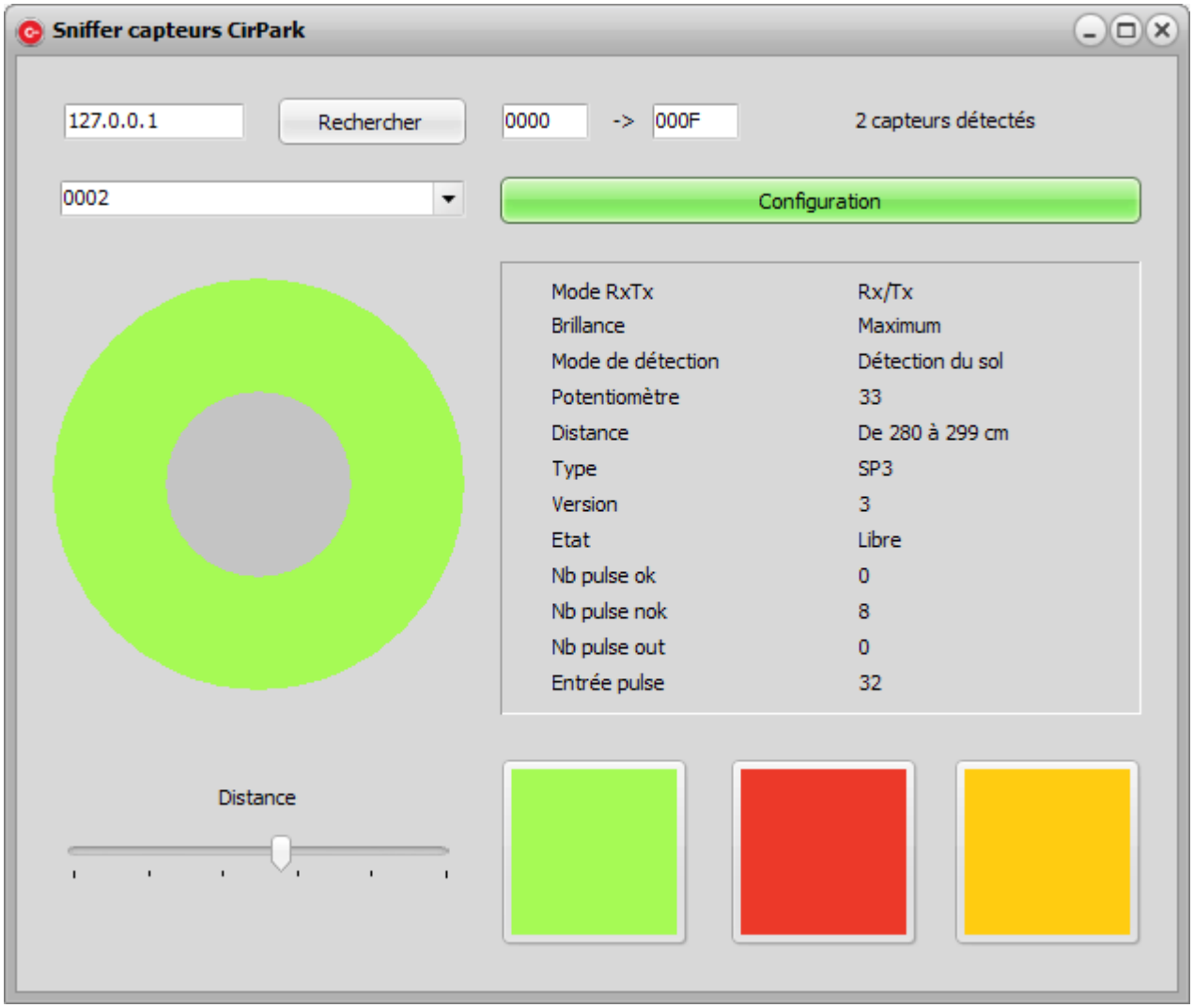
TP
Modification des paramètres des capteurs

La méthode EnvoyerRequeteCapteur de la classe IRSnifferCirpark permet de modifier la configuration d'un capteur, et notamment de changer sa couleur ou bien sa distance de détection.

Quels codes fonction permet de changer la couleur du capteur ?

Quel code fonction permet de modifier la distance de détection ?

Mettre en place l'interface finale : les boutons colorés sont des SpeedButton.



> Coder l'envoi des 4 requêtes (vert, rouge, orange, distance).

On donne :

```
if(TrackBar1->Position<160)donnee=0x08;
    else if(TrackBar1->Position<180)donnee=0x09;
    else if(TrackBar1->Position<190)donnee=0x0A;
    else if(TrackBar1->Position<210)donnee=0x0B;
    else if(TrackBar1->Position<230)donnee=0x0C;
    else if(TrackBar1->Position<250)donnee=0x0D;
    else if(TrackBar1->Position<270)donnee=0x0E;
    else if(TrackBar1->Position<280)donnee=0x0F;
    else if(TrackBar1->Position<300)donnee=0x10;
    else if(TrackBar1->Position<320)donnee=0x11;
    else if(TrackBar1->Position<340)donnee=0x12;
    else if(TrackBar1->Position<360)donnee=0x13;
    else if(TrackBar1->Position<380)donnee=0x14;
    else if(TrackBar1->Position<400)donnee=0x15;
    else donnee=0x16;
```

> Versionner le code complet.

Diagramme de classe complet