MODULE 02

Gestion de l'éclairage dans une salle de spectacle

Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

MODULE 02
SÉANCE SYSTÈME 01
TP D'INFORMATIQUE
Durée 2h30

CONFIGURATION DMX ET ANALYSE WIRESHARK

BLOC DE COMPÉTENCES
U5 - EXPLOITATION ET MAINTENANCE DE RÉSEAUX INFORMATIQUES

COMPÉTENCE(S)
C09 - INSTALLER UN RÉSEAU INFORMATIQUE

OBJECTIF PÉDAGOGIQUE
Installation et configuration de matériel DMX : analyse Wireshark des données.

CONNAISSANCES ISSUES DU RÉFÉRENTIEL

• Réseaux de terrain	Niveau 2
• Réseaux informatiques (protocoles, équipements et outils usuels et industriels)	Niveau 3
• Outils logiciels d'évaluation, de traçabilité de l'information	Niveau 3

CONNAISSANCES OPÉRATIONNALISÉES

• Installer et configurer un matériel à partir d'une documentation	Niveau 3
• Analyser une communication réseau avec Wireshark	Niveau 2

Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

TD
Étude de la norme et de la documentation du matériel

A partir de la norme DMX, donner la composition d'une trame DMX :

Quelles sont les durées maximum et minimum d'une trame DMX ?

D'après la documentation du SPOTEX15 (version 5 canaux), donner la composition de la trame permettant les effets suivants :

PAN à mis parcours, TILT au maximum, clignotement rapide d'une fleur rose :

PAN au minimum, TILT à mis parcours, motif "éclairs" Arc-en-ciel avec 50% de lumière :

En étudiant la documentation de 4 appareils DMX disponibles dans la salle, compléter le tableau :

Appareil DMX	Nombre de canaux utilisés
- SPOT EX-15	- 5 ou 13 canaux
-	-
-	-
-	-
-	-

En étudiant la documentation du matériel DMX qui vous est confié, remplir la fiche suivante en se limitant à 8 effets possibles :

Référence de l'appareil DMX :

Canal	Valeur	Effet
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-

TP
Installation du module Artnet-DMX Enttec, configuration du matériel DMX et analyse réseau des données transmises

Installation du module Enttec

Installer NMU (Node Management Utility).

Dans le logiciel NMU, lancer la recherche des modules Enttec présents sur le réseau, et compléter les informations suivantes (accessibles dans la fenêtre de configuration d'un module) :

- IP : Univers :
- IP : Univers :

Configuration de votre appareil DMX

En suivant attentivement la documentation constructeur de votre appareil :

- Configurez l'appareil en mode DMX.
- Affectez lui l'adresse DMX 10*numéroDeVotrePC (canal de départ)

Test de pilotage via NMU (module Enttec)

Choisir Arnet-test et valider le mode Arnet : vous pouvez maintenant actionner les curseurs et vérifier la bonne configuration de votre matériel en vérifiant les effets lumineux : attention chaque binôme/trinôme devra faire ses tests lorsque les autres n'en feront pas.

Analyse Wirshark des données

Quel filtre permet d'isoler les trames issues de votre PC et utilisant le protocole UDP ?

Lancer Wireshark et isoler les trames UDP issues de votre PC : indiquer l'entête de cette trame :

Indiquer, d'après la documentation constructeur du module Enttec, la signification des 14 premiers octets de donnée.

Vérifier l'évolution de la valeur des données correspondant à votre matériel lors du pilotage.

MODULE 02
SÉANCE SYSTÈME 02
TP D'INFORMATIQUE
Durée 2h30

CLIENT TCP EN C++ : ENVOI TRAME DMX

BLOC DE COMPÉTENCES
U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)
C08 - CODER

OBJECTIFS PÉDAGOGIQUES
Analyse UML de l'installation. Codage d'un client TCP permettant de mettre les appareils d'éclairage en mouvement.

CONNAISSANCES ISSUES DU RÉFÉRENTIEL

• Langages de modélisation	Niveau 2
• Programmation orientée objet	Niveau 3
• Programmation réseau	Niveau 2
• Outils logiciels d'évaluation, de traçabilité de l'information	Niveau 3

CONNAISSANCES OPÉRATIONNALISÉES

• Utiliser une classe C++	Niveau 3
• Envoyer une trame TCP	Niveau 3
• Analyser une communication réseau avec Wireshark	Niveau 2
• Versionner un code	Niveau 2

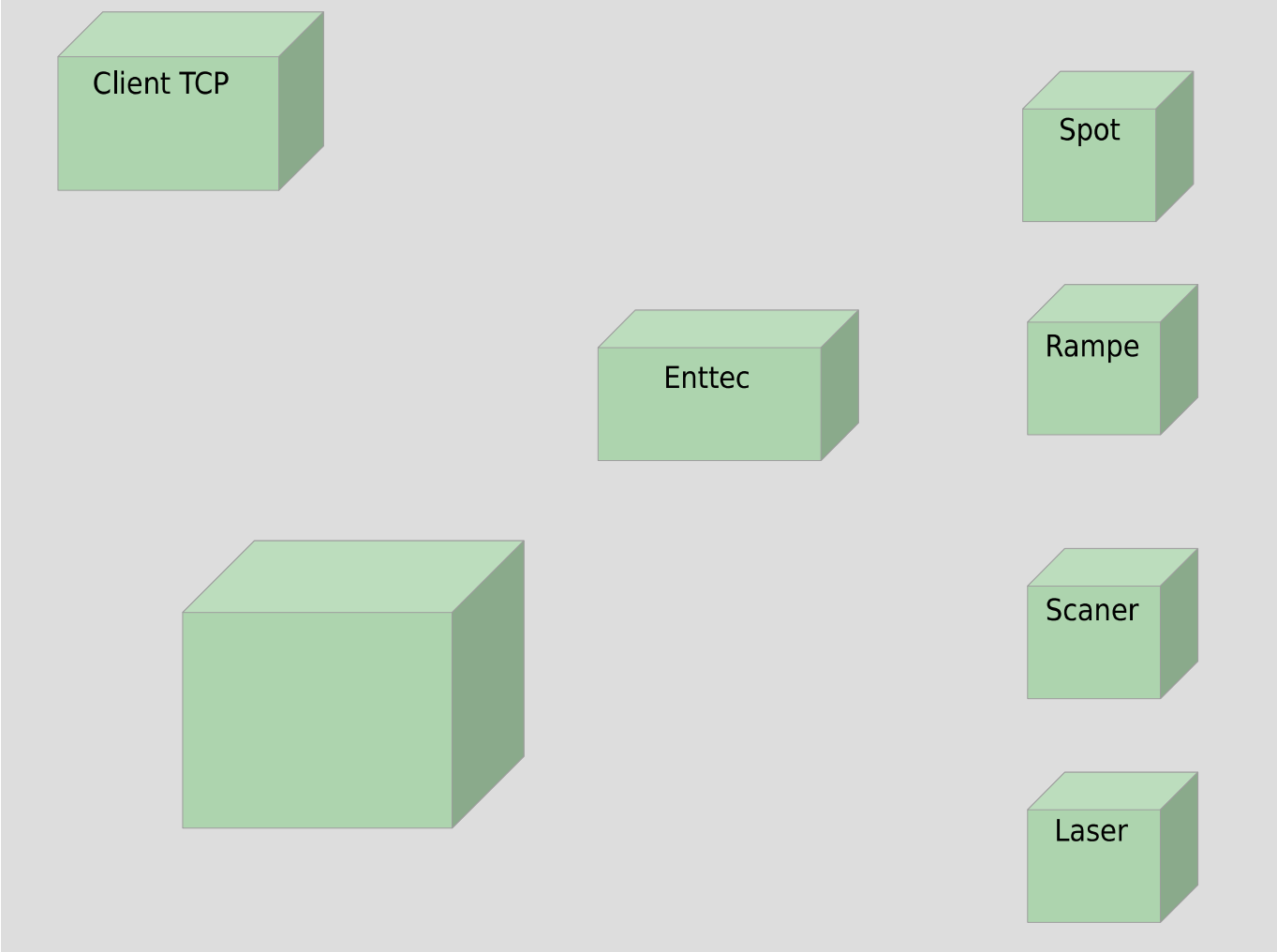
Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

TD
Analyse UML : diagramme de déploiement

Le module Enttec est piloté en UDP, il génère les trames DMX permettant de piloter le système d'éclairage de scène (les appareils DMX). Un serveur TCP/IP permet de convertir les trames TCP, provenant des applications clientes, en trame UDP pour le module Enttec : il permet de sauvegarder l'historique des trames envoyées. Ce serveur TCP/IP est donc aussi un client UDP.

Compléter le diagramme de déploiement du système, en précisant les protocoles mis en jeu :



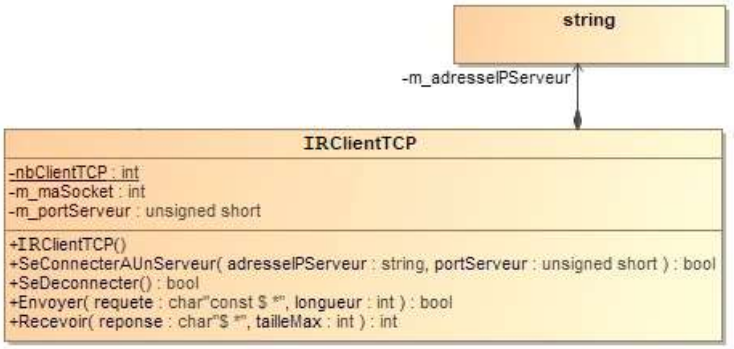
Quelles sont les différences entre un client et un serveur ?

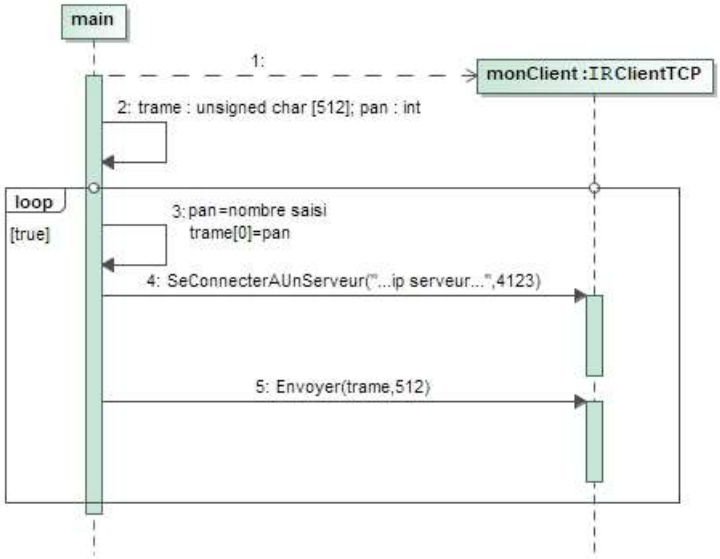
Quelle est la différence entre UDP et TCP ?

TP
Envoi d'une trame fixe en mode console : client TCP

Le boîtier Enttec est piloté en UDP, un serveur TCP/IP (serveurTCPClientEnttec.exe) permet de convertir les trames TCP en trame UDP pour le boîtier : c'est aussi un client UDP.

La classe IRClientTCP est donnée :





> Coder le diagramme de séquence donné, afin d'envoyer la trame de votre choix (vérifier que la ligne #define WIN32_LEAN_AND_MEAN est bien présente dans IRClientTCP.h). Votre code :

> Vérifier les effets lumineux produits.

> BONUS : proposer à l'utilisateur de saisir plusieurs effets DMX, modifier et envoyer a trame en fonction des effets suivis.

> Versionner le code complet.

TP
Analyse Wireshark des communications TCP et UDP

Pour analyser à l'aide de WireShark la trame TCP envoyée au serveur TCP ainsi que la trame UDP, il faut lancer le serveur TCP et le client TCP sur 2 postes différents. L'analyse Wireshark doit être lancée sur le poste serveur TCP. Le filtre à utiliser est (il faut modifier les adresses) :

```
ip.addr == 172.20.182.229 || (ip.addr == 172.20.100.245 && tcp) || (ip.addr == 172.20.101.1 && tcp)
172.20.182.229 est l'adresse du ENTTEC (serveur UDP)
172.20.100.245 est l'adresse du serveur TCP / Client UDP
172.20.101.1 est l'adresse du client TCP (votre logiciel client TCP)
```

> Lancer l'analyse et expliquer les trames en détail, préciser les adresses mac et les ports

> BONUS : en se basant sur les relevés WireShark, proposer un diagramme UML de séquence visant à décrire (lors de l'envoi d'un effet) les échanges entre le client TCP, la passerelle TCP/UDP, le module Ethernet/DMX, et l'appareil DMX.

MODULE 02

SÉANCE SYSTÈME 03

TP D'INFORMATIQUE

Durée 2h30

CLASSE DMX EN C++

BLOC DE COMPÉTENCES

U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)

C08 - CODER

OBJECTIFS PÉDAGOGIQUES

Codage d'une classe DMX en C++, ajout des fonctionnalités : fullON, FullOFF, démonstration.

CONNAISSANCES ISSUES DU RÉFÉRENTIEL

- Langages de modélisation d'application — Niveau 3
- Programmation orientée objet — Niveau 3
- Programmation réseau — Niveau 3

CONNAISSANCES OPÉRATIONNALISÉES

- Écrire une classe C++ à partir d'un diagramme de classes — Niveau 2
- Versionner un code — Niveau 2

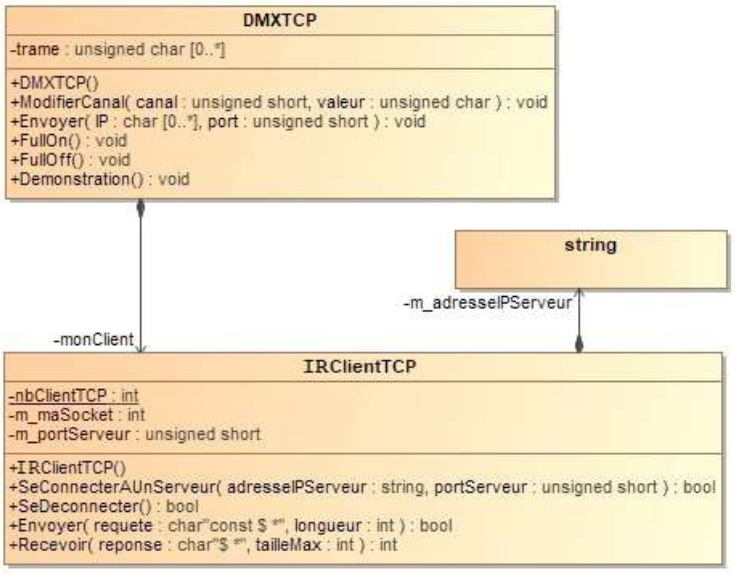
Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

TD
Déclaration d'une classe en C++ et définition des méthodes

Déclaration de la classe DMXTCP dans le fichier DMXTCP.h

La classe DMXTCP est composée d'un objet privé monClient de la classe IRClientTCP : cet objet est un attribut de la classe DMXTCP.



Lister les attributs de la classe DMXTCP :

Lister les noms des méthodes de cette classe :

Préciser le nom du constructeur :

Compléter la déclaration de la classe DMXTCP, en étudiant le diagramme de classe ci-dessus :

```
class DMXTCP
{  private :
   unsigned char trame[512] ;
   …                                  //créer l'objet monClient
   public :
   DMXTCP() ;
   void ModifierCanal(unsigned short canal, unsigned short valeur) ;
   void Envoyer(char IP[16], unsigned short port) ;
   …                                  //déclarer la méthode FullOn()
   …                                  //déclarer la méthode FullOff()
   …                                  //déclarer la méthode Demonstration()
} ;
```

Quelle est l'incidence des champs privé et publique dans l'utilisation d'une classe ?

Quel est le symbole géométrique d'une composition ?

Définition des méthodes de la classe DMXTCP dans le fichier DMXTCP.cpp

Compléter la définition du constructeur permettant de mettre à zéro les 512 octets de la trame :

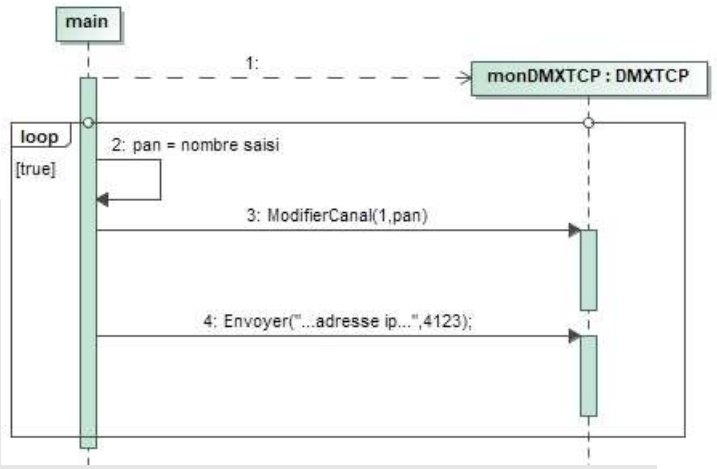
```
DMXTCP::DMXTCP
{  for(int i=0;i<512;i++)
   {    trame[i]=…
   }
}
```

Compléter la définition de la méthode ModifierCanal permettant de mettre valeur dans la case d'indice canal-1 du tableau d'octets trame :

```
void DMXTCP::ModifierCanal(unsigned short canal, unsigned short valeur)
{  trame[……]=…….. ;
}
```

Écrire la définition de la méthode Envoyer(char IP[16], unsigned short port) permettant de connecter monClient, d'envoyer le tableau trame, puis de se déconnecter :

Écrire le programme principal permettant de créer un objet monDMXTCP de la classe DMXTCP, de demander la saisie de la valeur du premier canal, d'appeler la méthode ModifierCanal, puis la méthode Envoyer.




```
Int main()
{
   return 0 ;
}
```

TP
Codage et test de la classe DMXTCP

Coder la classe DMXTCP : déclaration de la classe dans un fichier DMXTCP.h, définition des méthodes dans un fichier DMXTCP.cpp. Ajouter les classes nécessaires dans un nouveau projet et tester le programme principal écrit dans le TD.

Vérifier l'effet sur l'appareil d'éclairage de scène.

Coder les méthodes FullOff() (les 512 octets sont mis à 0) et FullOn() (les 512 octets sont mis à 255). Le code de FullOff() :

Coder la méthode Demonstration() permettant de générer 512 octets aléatoires : srand(time(0)); permet d'initialiser la fonction random (aléatoire). rand() renvoie un nombre aléatoire (à stocké dans chaque octet de la trame). Votre code :

BONUS : Coder l'interface console complète, en ajoutant le menu suivant :

MANUEL (1), FULLON (2), FULLOFF (3), DEMO (4)

MANUEL permet de saisir manuellement 2 effets au moins, DEMO permet de rentrer dans une boucle infinie while(true){...} cadencée par la fonction Sleep(1000) pour une temporisation de 1000ms. Le code du programme principal complet :

Versionner le code complet.

MODULE 02

SÉANCE SYSTÈME 04

TP D'INFORMATIQUE

Durée 2h30

INTERFACE GRAPHIQUE : CONSOLE VIRTUELLE

BLOC DE COMPÉTENCES

U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)

C08 - CODER

OBJECTIFS PÉDAGOGIQUES

Codage d'une interface graphique munie de curseurs (scrollbar) permettant de piloter des appareils d'éclairage de scène.

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

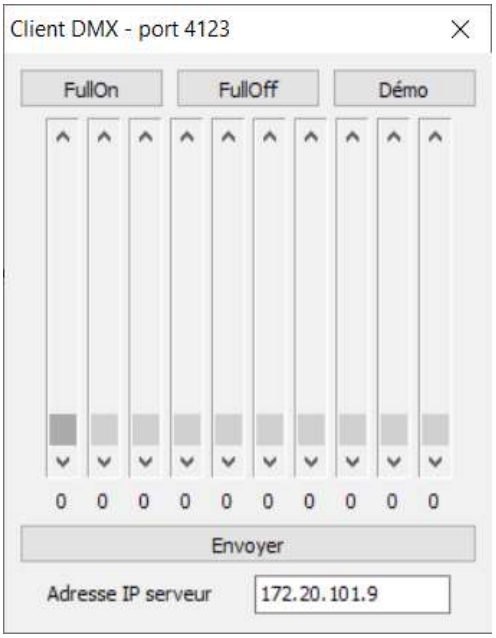
- Concevoir une interface graphique sous Windows — Niveau 2
- Versionner un code — Niveau 2

Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

TP
Console virtuelle simple : interface graphique

Une vidéo-tuto est disponible.



Créer une nouvelle application graphique sous C++ Builder : Placer Buttons, Labels, ScrollBars (mode vertical) et Edit dans l'interface. Dans le code source Unit1.h, ajouter un objet monDMX de la classe DMXTCP, dans les attributs privés de la classe TForm1 (la classe qui gère l'IHM - Interface Homme Machine), penser aux #include. Sauvegarder tout et compiler le projet.

Générer l'événement OnChange de l'objet ScrollBar. Dans le code de cet événement : affecter à la propriété Caption de chaque Label : (255 - position du ScrollBar) - par défaut la valeur maximum du ScrollBar est en bas -. Pour accéder à une propriété taper Label1-> (pour le Label1), ScrollBar1-> (pour le ScrollBar1) : un menu déroulant s'affiche. Sauvegarder et compiler le projet.

Noter le code complet (avec entête) de l'événement OnChange de l'objet ScrollBar1 :

Utiliser l'objet monDMX afin de lancer la méthode ModifierCanal : passer en argument le numéro du canal et la valeur (255 - ScrollBar1 ->Position, pour le premier canal). monDMX. permet d'accéder aux méthodes de l'objet monDMX. Sauvegarder et compiler le projet.

A chaque modification du code, il faut construire le projet (SHIFT F9) et résoudre les erreurs. Dans le cas contraire, les menus déroulants ne s'affichent plus, le codage est ralenti et perd en efficacité, les erreurs s'accumulent…

Générer l'événement OnClick du bouton Envoyer. Utiliser l'objet monDMX afin de lancer la méthode Envoyer en passant en argument l'adresse IP saisie dans la zone Edit (une conversion est nécessaire, il faut saisir à la place de l'adresse : AnsiString(Edit1 ->Text).c_str(), puis le numéro du port (4123). Testez les fonctionnalités du logiciel que vous venez de créer.

Noter le code complet (avec entête) de l'événement OnClick de l'objet bouton Envoyer :

Modes FullOff, FullOn : coder les événements associés aux boutons FullOn / FullOff en appelant les bonnes méthodes à l'aide le l'objet monDMX. Tester les fonctionnalités du logiciel.

Le code complet (avec entête) de l'événement OnClick de l'objet bouton FullOff :

BONUS : Dans les options du projets : rechercher Apparence, vous pouvez modifier le « skin » de votre interface et ajouter un icône.

Votre code sera indépendant du compilateur si vous désactivez RTL dynamique dans le linker (Décocher : Link with Dynamic RTL), et si vous désactivez les packages (Décocher : Link with runtime packages)

Versionner le code complet.

BONUS : Mode Démonstration : ajouter à l'interface un objet TTimer. Dans ses propriétés : Enabled=false. Dans l'événement OnTimer appeler la méthode Demonstration() de l'objet monDMX. Appeler ensuite la méthode Envoyer() de l'objet monDMX, comme précédemment.

Le code complet (avec entête) de l'événement OnTimer de l'objet Timer1 :

```

```

Codage du bouton Démo : activer le timer en fonction de la valeur du Caption du bouton :

```
Si la propriété Caption du bouton est "Démo" : la propiété Enabled du timer = true, le Caption du bouton devient "Stop".
Si la propriété Caption du bouton est "Stop" : la propiété Enabled du timer = false, le Caption du bouton devient "Démo".
```

Remarque : les objets Captions sont de type UnicodeString, ils acceptent la comparaison directe à une chaîne de caractères à l'aide de l'opérateur == (contrairement aux tableaux de caractères qui nécessitent l'utilisation de la fonction strcmp()).

Le code complet de l'événement OnClick de l'objet bouton Démo :

```

```

BONUS : Dans les options du projets : rechercher Apparence, vous pouvez modifier le « skin » de votre interface et ajouter un icône.

Votre code sera indépendant du compilateur si vous désactivez RTL dynamique dans le linker (Décocher : Link with Dynamic RTL), et si vous désactivez les packages (Décocher : Link with runtime packages)

Versionner le code complet.

MODULE 02

SÉANCE SYSTÈME 05

TP D'INFORMATIQUE

Durée 2h30

INTERFACE GRAPHIQUE : BOUTONS RAPIDES

BLOC DE COMPÉTENCES

U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)

C08 - CODER

OBJECTIFS PÉDAGOGIQUES

Codage d'une interface graphique spécialisée pour un matériel d'éclairage : boutons d'accès rapide aux effets lumineux.

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

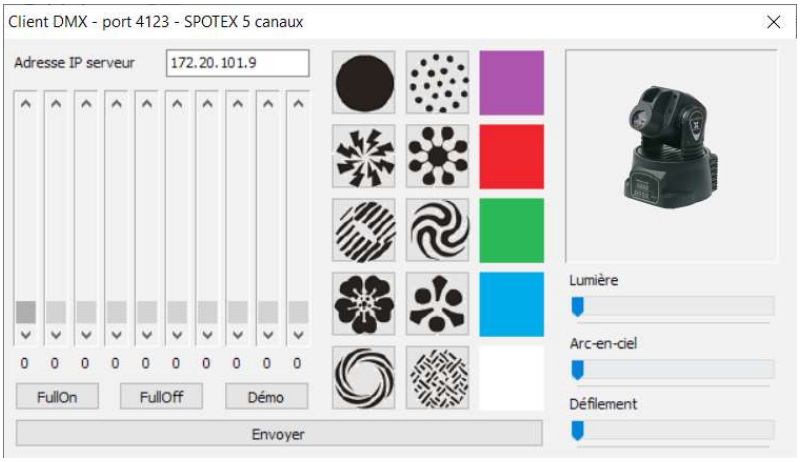
Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

TP
Interface spécialisée

Il est demander de modifier l'interface « console virtuelle » codée et testée précédemment, afin d'y ajouter des fonctionnalités de commande rapides pour le SPOTEX15.

Les boutons pourvus d'images sont des SpeedButtons (proprété Glyph pour l'image .bmp), les curseurs -Lumière, Arc-en-ciel, Défilement- sont des TrackBars.



Chaque SpeedButton doit modifier l'attribut Position des ScrollBars en fonction de l'effet demandé : le code doit être placé dans l'événement OnClick du bouton. Le déplacement des TrackBars doit également modifier l'attribut Position des ScrollBars en fonction de l'effet demandé : le code doit être placé dans l'événement OnChange du TrackBar.

> Coder et tester les fonctionnalités des boutons et des TrackBars. Chaque événement appellera l'événement OnClic du bouton Envoyer, l'effet sera ainsi immédiat.

> Le code complet de l'événement OnClick correspondant au bouton « rouge » :

> Le code complet de l'événement OnChange de l'objet TrackBar1 (Lumière) :

> BONUS : ajouter un bouton A propos de... permettant d'afficher dans une nouvelle fenêtre les caractéristiques du SPOTEX15

> Versionner le code complet.

MODULE 02

SÉANCE SYSTÈME 06

TP D'INFORMATIQUE

Durée 2h30

INTERFACE GRAPHIQUE : JOYSTICK VIRTUEL

BLOC DE COMPÉTENCES

U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)

C08 - CODER

OBJECTIFS PÉDAGOGIQUES

Codage d'un joystick virtuel permettant de mettre en mouvement matériel d'éclairage.

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

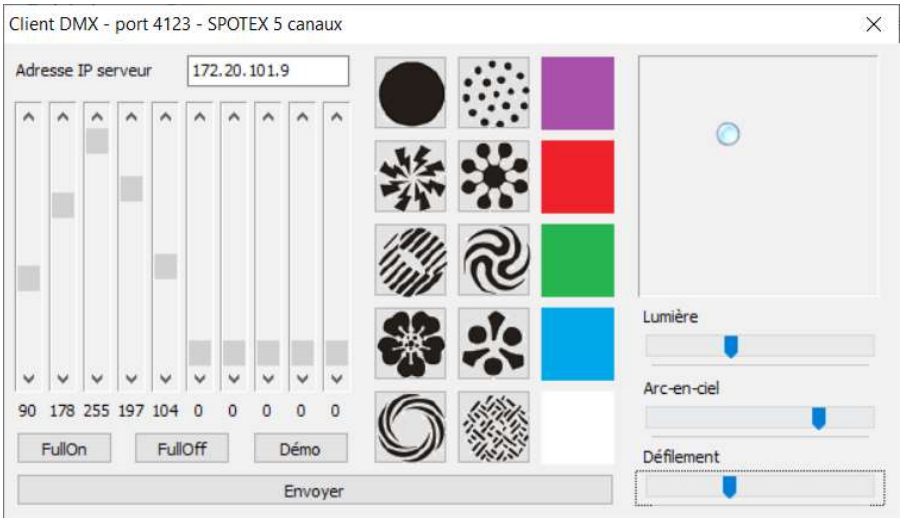
- Gestion des événements souris sur un objet graphique. — Niveau 2
- Versionner un code — Niveau 2

Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

TP
Gestion des événements souris : joystick virtuel

Le "trackBall" virtuel est une image (TImage) dans un Bevel.



Ajouter à TForm1 l'attribut click, booléen. Les événements du TImage à utiliser sont : OnMouseDown (click est positionné à true), OnMouseUp (click est positionné à false) et OnMouseMove. L'événement OnMouseMove fournit les coordonnées X et Y du déplacement de la souris sur l'image, il faut utiliser ces 2 valeurs pour déplacer l'image en conséquence et modifier la position des ScrollBars correspondant au PAN et au TILT (à condition que click soit true). Plusieurs tests doivent être ajoutés pour que l'image ne sorte pas du Bevel.

Tester les fonctionnalités du logiciel spécialisé que vous venez de créer.

Le code complet de l'événement OnMouseMove :

BONUS : en utilisant la trigonométrie, modifier le code afin que les mouvements du faisceau lumineux sur un plan vertical suivent exactement les mouvements du joystick virtuel.

Versionner le code complet.

MODULE 02

SÉANCE SYSTÈME 07

TP D'INFORMATIQUE

Durée 2h30

ANDROID : ENVOI TRAME UDP

BLOC DE COMPÉTENCES

U5 - EXPLOITATION ET MAINTENANCE DE RÉSEAUX INFORMATIQUES

COMPÉTENCE(S)

C06 - VALIDER UN SYSTÈME INFORMATIQUE

OBJECTIFS PÉDAGOGIQUES

Installation Android Studio et test de déploiement sur tablette. Codage de l'envoi d'une trame UDP permettant de mettre en mouvement un appareil d'éclairage.

CONNAISSANCES ISSUES DU RÉFÉRENTIEL

• Maîtrise des environnements de développement, d'intégration, de déploiement logiciel et des versions logicielles associées	Niveau 2
• Langages de développement, de description, de création d'API et les IDE associés	Niveau 3
• Tests unitaires et d'intégration	Niveau 3

CONNAISSANCES OPÉRATIONNALISÉES

• Installer un environnement de développement et de test pour appareil mobile sous Android	Niveau 2
• Coder une application pour Android utilisant les sockets	
• Versionner un code	Niveau 2

Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

TP
Installation Android Studio et test de déploiement sur tablette

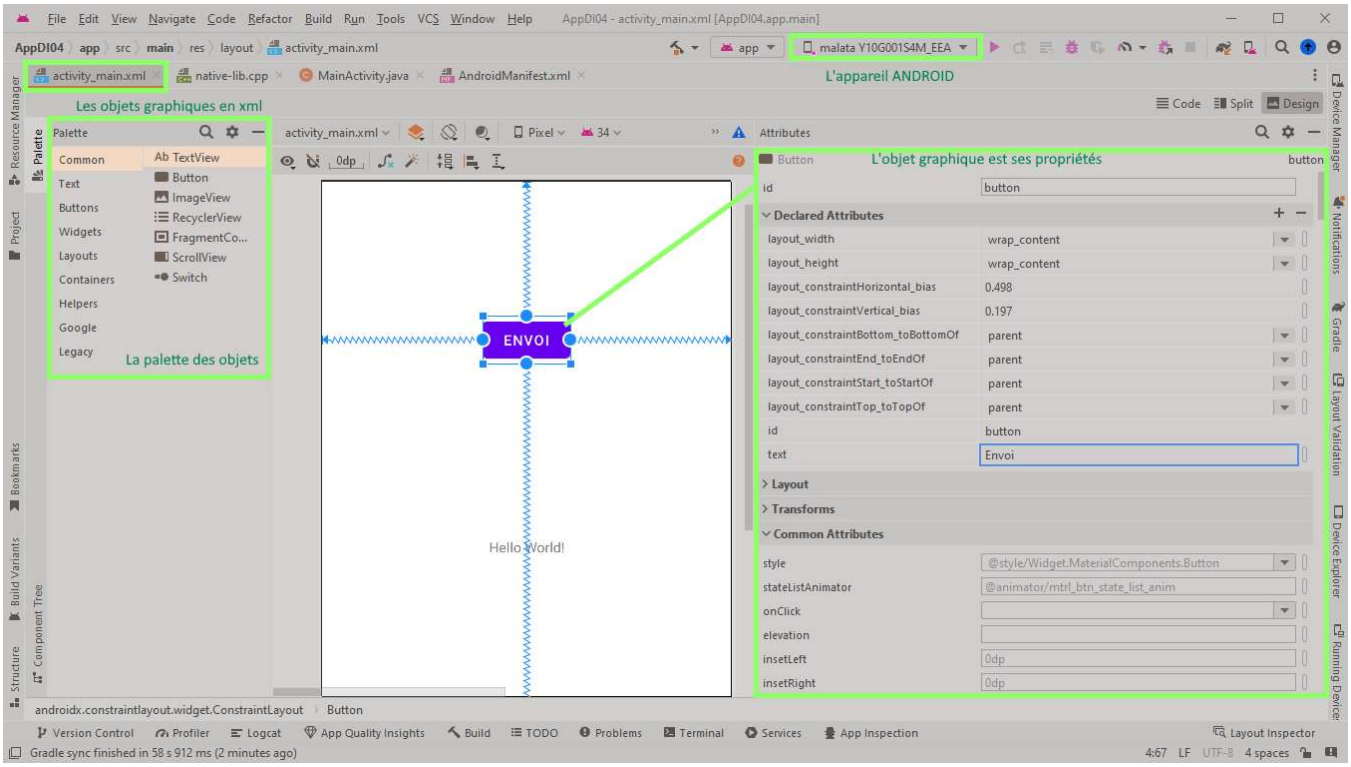
> Installer android-studio-2022.3.1.19-windows.exe.
>
> Créer un nouveau projet en choisissant Native-C++ dans sa version Java pour l'interface : penser à donner un nom pertinent à votre projet en le créant. Compiler et déployer le programme de base Hello World : la tablette doit être en mode développeur, elle doit accepter le mode debug lors de sa connexion au PC de programmation.

3 fichiers nous intéressent :

activity_main.xml

Contient l'interface graphique :

- mode Code : l'interface en xml qui liste les objets graphiques et leurs propriétés.
- mode Design : permet de placer graphiquement les objets dans l'interface et de visualiser leurs propriétés.
- mode Split : permet de visualiser en même temps le code et le design.



native-lib.cpp

Contient le code C++ : c'est en C++ que la communication avec le serveur sera gérée.

MainActivity.java

Contient la gestion des événements et les méthodes d'interaction avec le code C++.

TP
Android studio en native-C++
Envoi d'une trame fixe TCP permettant de générer les effets lumineux

> Ajouter un bouton dans l'interface : lors du click, une trame TCP sera envoyée au serveur DMX.

native-lib.cpp :

```
#include <jni.h>
#include <string>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_appdi04_MainActivity_stringFromJNI( //appdi04 est le nom de l'application
    JNIEnv* env,
    jobject /* this */) {
    std::string message;
    std::string m_adresseIPServeur = "172.20.15.34"; // à modifier
    int m_portServeur = 17777;                       // à modifier
    int m_maSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(m_maSocket == -1)
    {   message="Creation de la socket : ERREUR.";
        return env->NewStringUTF(message.c_str());
    }
    message="Creation de la socket : OK.";
    struct sockaddr_in serveurAJoindre;
    serveurAJoindre.sin_family = AF_INET;
    serveurAJoindre.sin_addr.s_addr = inet_addr(m_adresseIPServeur.c_str());
    serveurAJoindre.sin_port = htons(m_portServeur);
    int resultat = connect(m_maSocket, (const struct sockaddr *) &serveurAJoindre,
sizeof(serveurAJoindre));
    if(resultat != 0) {
        message = "Connexion au serveur : ERREUR.";
        return env->NewStringUTF(message.c_str());
    }
    message= message+"\nconnecté au serveur";
    std::string messageAEnvoyer="message du client";  // sera remplacer par les 512 octets de la
trame DMX : stockés dans un tableau de char
    resultat = send(m_maSocket, messageAEnvoyer.c_str(), messageAEnvoyer.length(), 0);
    if(resultat == -1)
    {   message = message+"\nEnvoi du message : ERREUR.";
        return env->NewStringUTF(message.c_str());
    }
    return env->NewStringUTF(message.c_str());
}
```

MainActivity.java :

Ici appdi04 est le nom de l'application : attention, une partie du code est auto-générée. Il faut absolument utiliser l'autocomplétion du code. Dans le code ci-dessous, sample_text et l'Id du Label, button est l'Id du bouton.

```
package com.example.appdi04;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import com.example.appdi04.databinding.ActivityMainBinding;
public class MainActivity extends AppCompatActivity {
    static {
        System.loadLibrary("appdi04");
    }
    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        TextView text=findViewById(R.id.sample_text);
        text.setText("clicker");
        Button button1=(Button) findViewById(R.id.button);
        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                text.setText("bravo");
            text.setText(stringFromJNI()); //appel de la méthode C++
            }
        });
    }
    public native String stringFromJNI();
}
```

> Tester le code en utilisant un serveur TCP de test (par exemple : TestClientServeurTCPUDP V5.exe)

> Modifier dans le code C++ le message envoyé au serveur afin de lui envoyer les 512 char de la trame DMX.

> Vérifier les effets lumineux.

> Versionner le code complet.

MODULE 02

SÉANCE SYSTÈME 08

TP D'INFORMATIQUE

Durée 2h30

ANDROID : CONSOLE VIRTUELLE

BLOC DE COMPÉTENCES

U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)

C08 - CODER

OBJECTIFS PÉDAGOGIQUES

Codage d'une application Android munie de curseurs (seekbar) permettant de piloter des appareils d'éclairage de scène.

CONNAISSANCES ISSUES DU RÉFÉRENTIEL

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

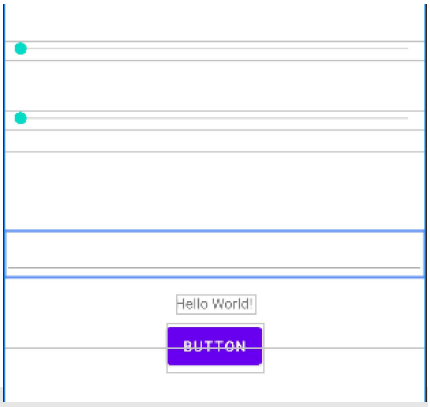
CONNAISSANCES OPÉRATIONNALISÉES

- Concevoir une interface graphique sous Android : gestion des événements java et communication en C++ natif — Niveau 2
- Versionner un code — Niveau 2

Gaëtan GEORGES (g2.iris.lla@gmail.com)
Pascal HAGOT (pascal.hagot@ac-creteil.fr)
David IERVESE (d_iervese@yahoo.com)

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

TP
Android studio en native-C++
Gestion des seekbar : console DMX virtuelle



Gestion des seekbar et envoi d'un tableau de caractères de l'interface Java au code C++ natif :

> Placer 2 seekbar, 1 Edit, 1 label et 1 boutons dans l'interface, et tester le code suivant.

MainActivity.java :

```
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import com.example.myapplication.databinding.ActivityMainBinding;
import com.google.android.material.textfield.TextInputEditText;
public class MainActivity extends AppCompatActivity {
    static {        System.loadLibrary("myapplication");    }
    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        Button button1=(Button) findViewById(R.id.button);
        TextInputEditText text=(TextInputEditText)findViewById(R.id.edit);
        SeekBar seekbar=(SeekBar)findViewById(R.id.seekBar);
        SeekBar seekbar2=(SeekBar)findViewById(R.id.seekBar2);
        int [] trameDMX= {1,2,0,0,0,0,0,0,0,0};
        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                TextView tv = binding.sampleText;
                tv.setText(stringFromJNI(text.getText().toString(),trameDMX));
            }
        });
        seekbar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                trameDMX[0]=seekBar.getProgress();
                binding.sampleText.setText(String.valueOf(trameDMX[0]));
            }
        });
        seekbar2.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                trameDMX[1]=seekBar.getProgress();
                binding.sampleText.setText(String.valueOf(trameDMX[1]));
            }
        });
    }    public native String stringFromJNI(String text, int [] trame);
}
```

native-lib.cpp :

```
#include <jni.h>
#include <string>
#include <string.h>
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_myapplication_MainActivity_stringFromJNI(
        JNIEnv* env,
        jobject /* this */,
        jstring edit, jintArray trame) {
    int *tableau=env->GetIntArrayElements(trame,NULL);
    const char *chaine=env->GetStringUTFChars(edit,NULL);
    sprintf((char*)chaine,"%s %d %d",chaine,tableau[0],tableau[1]);
    return env->NewStringUTF(chaine);
}
```

> Modifier le code C++ en intégrant le code du client TCP : le tableau trame sera envoyé au serveur.

> BONUS : modifier le code afin de placer 6 seekbars. Vérifier les effets lumineux.

> Versionner le code complet.