Colbert

Fresi

Wantelez

Compte Rendu TP : Camera

SN2 - DOSSIER N°2

Système : Vidéo surveillance

Fiche résumée

Le système vidéo surveillance est séparé en deux parties :

La caméra 2 axes pilotables en local via la liaison RS232

La carte d'acquisition vidéo.

Nous ne nous intéressons ici qu'au module de pilotage via la liaison série de la caméra. Votre application doit permettre de piloter tous les axes de la caméra ainsi que le Zoom interne de celle-ci. Un écran vidéo portable permettra de contrôler les points visés par la caméra.

SOUS SYSTEME LIAISON SERIE

TP 2 :

Acquisitions fondamentales

Etude globale du système

Etude de la liaison série et du langage propriétaire Sony

Analyse et validation des savoirs

Envoi d'une trame d'initialisation à la caméra via RS232

Recherche et synthèse

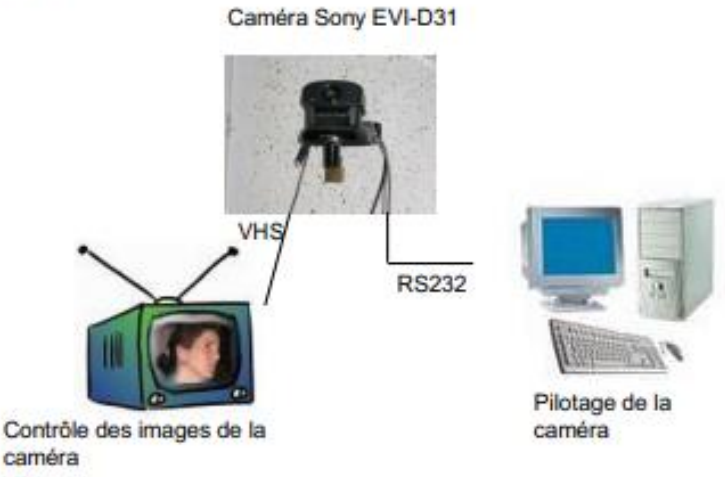
Recherche d'un mode de fonctionnement automatique de la caméra.

Codage et tests.

Documents fournis :

Documentation technique de la caméra

API win32 (ftp en cours dans la section)



GRILLE DE NOTATION – TP SYSTEMES

NOMS DES ETUDIANTS		Application vérifiée par :	Date
TITRE DU TP :			

QUESTIONS PRELIMINAIRES	**NOTATION**	**/ 4**
Précision et pertinence des réponses	/4	
Qualité de la rédaction (présentation, orthographe, grammaire,...)	Bonus 1pt	

COMPTE RENDU	**NOTATION**	**/ 6**
Sommaire, but (obligatoire sinon perte de points)	-1 pt possible	
Principe	/3	
Copies d'écrans avec explications(obligatoire sinon perte de points)	-1 pt possible	
Conclusion (problèmes rencontrés, résolution, vécu du TP, proposition d'évolution)	/2	
Qualité de la rédaction du compte rendu	/1	

APPLICATION	**NOTATION**	**/ 10**
Classe(s), (Réutilisabilité, complétude, Utilisation, syntaxe)	/3	
Qualité du code (entête et commentaires, Qualité d'implémentation)	/3	
Correspondance avec les objectifs (recettage)	/3	
Capacité à réaliser tout leTP	/1	
	TOTAL /20	

COMMENTAIRES / REMARQUES / CONSEILS :

Sommaire

But

Le but de ce projet est de réussir à contrôler une caméra à distance à l'aide d'une application en C++. La caméra a une liaison en RS232, la caméra doit pouvoir se déplacer sur son axe vertical et horizontal mais aussi avoir la capacité de zoomer et de dézoomer à notre guise.

Principe

Etape 1 : Téléchargement et analyse des différents documents sur pearltrees en lien avec le TP.

Etape 2 : Analyser la doc de la caméra et trouver les trames utiles pour le projet.

Etape 3 : Téléchargement de Visual Studio Code 2017 et se renseigner sur le fonctionnement de l'API Win32.

Etape 4 : Création du projet sur Visual Studio Code et de la classe liaison.

Etape 5 : Création du Form pour récupérer contrôler la caméra.

Etape 6 : Créer les méthodes pour ouvrir le port et envoyer les trames.

Etape 7 : Appel des méthodes dans le main.

Etape 7 : Compilation du programme et tests des différentes méthodes.

Réponses aux questions spécifiques

1) Donnez les caractéristiques de la caméra et de ses possibilités de déplacements

La caméra EVI 31 de SONY est capable de zoomer, faire un « focus », changer la luminosité de l'image, ou encore bouger ver la gauche, droite, haut, bas, et en diagonale

2) Comment sont transmises les données sur la liaison série (expliquez le fonctionnement de la transmission) ?

Les données sont transmises sous formes de trame en RS232

3) Différence entre liaison synchrone et asynchrone ?

Une liaison synchrone nécessite que l'émetteur et le récepteur aient des horloges synchronisées, tandis qu'une liaison asynchrone ne nécessite aucune synchronisation

4) Donnez le format de transmission (vitesse ...) de la caméra ?

Le format de transmission est de 9600 bps

5) Que veut dire la chaîne de caractères (code ascii) suivante pour la caméra :0x81, 0x01, 0x04, 0x00, 0x03, 0xff ?

La trame signifie « off », pour éteindre la caméra

6) Donnez les chaînes Ascii qui permettent de :

-Allumer (Allumer la caméra)

On : 8x 01 04 00 02 FF

-Eteindre (Eteindre la caméra)

Off : 8x 01 04 00 03 FF

Algorithme :

```
//---------------------------------------------------------------------------

#pragma hdrstop

#include "Liaison.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#include <winsock.h>
#include <stdlib>
#include <stdio.h>


bool Liaison::OuvrirPort()    //on ouvre le port souhaite et on le configure
{
  this->hCom = CreateFileA("COM1",GENERIC_READ | GENERIC_WRITE,0,NULL,OPEN_EXISTING,FILE_FLAG_NO_BUFFERING,NULL);

  if(hCom == INVALID_HANDLE_VALUE)
  {
      return false;
  }
  else
  {
    DCB conf;

    GetCommState(this->hCom,&conf);

    conf.BaudRate = CBR_9600;
    conf.StopBits = ONESTOPBIT;
    conf.Parity = NOPARITY;
    conf.ByteSize=8;
    SetCommState(this->hCom,&conf);
    COMMTIMEOUTS comm;
    comm.ReadIntervalTimeout = MAXDWORD;
    comm.ReadTotalTimeoutMultiplier=0;
    comm.ReadTotalTimeoutConstant=0;
    comm.WriteTotalTimeoutMultiplier=0;
    comm.WriteTotalTimeoutConstant=0;

    SetCommTimeouts(hCom,&comm);

    return true;
  }
}
```

Ci-dessus, la configuration du port série pour permettre une communication entre le Form et la caméra. On retrouve le port dans CreateFile (dans notre cas le COM1), et la suite le nombre de baud, la nombre de bit, la parité et le bit de stop.

```
bool Liaison::sendMsg(char * buffer)
{
    bool etat = false;
    unsigned long tailleRead;

    etat = WriteFile(this->hCom,buffer,strlen(buffer),&tailleRead,NULL);

    if(etat == true)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

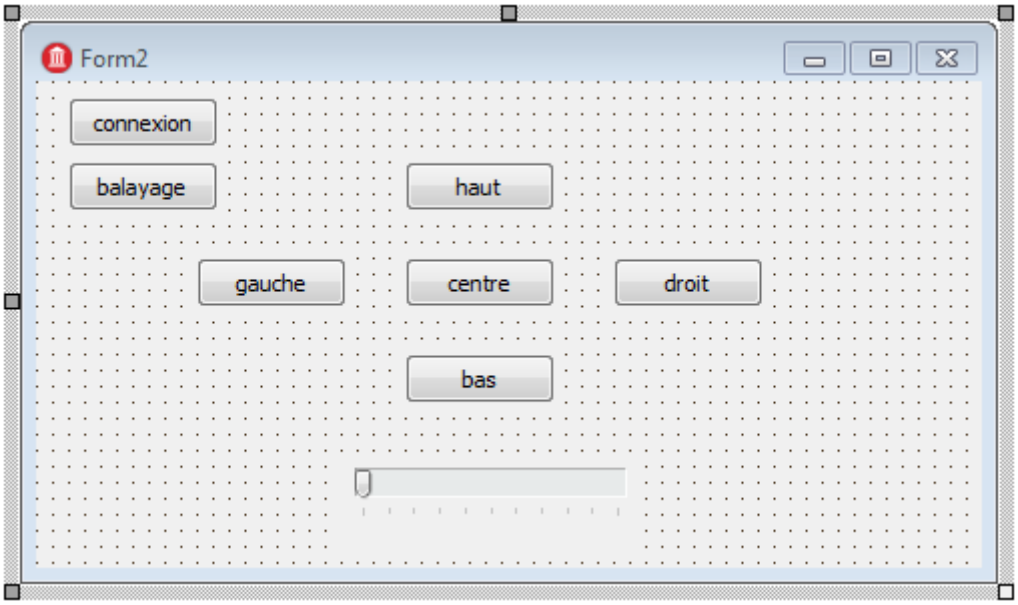
Ci-dessus, la fonction pour envoyer une trame à la caméra pour effectuer une action.

```
bool Liaison::fermerPort()
{
    bool etat;

    etat = CloseHandle(this->hCom);

    if(etat == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Fonction pour fermer le port (classique).



Ci-dessous, un exemple d'envoi d'une trame, dans notre cas pour que la caméra aille vers la droite.

```
//---------------------------------------------------------------------------
// Fonction qui envoie la trame pour faire une rotation vers la droite à la caméra.
void __fastcall TForm2::ButtonDroitClick(TObject *Sender)
{
    char trameD[9];

    trameD[0] = 0x81;
    trameD[1] = 0x01;
    trameD[2] = 0x06;
    trameD[3] = 0x01;
    trameD[4] = 0x18;
    trameD[5] = 0x14;
    trameD[6] = 0x02;
    trameD[7] = 0x03;
    trameD[8] = 0xFF;

    portserie.sendMsg(trameD);
}
```