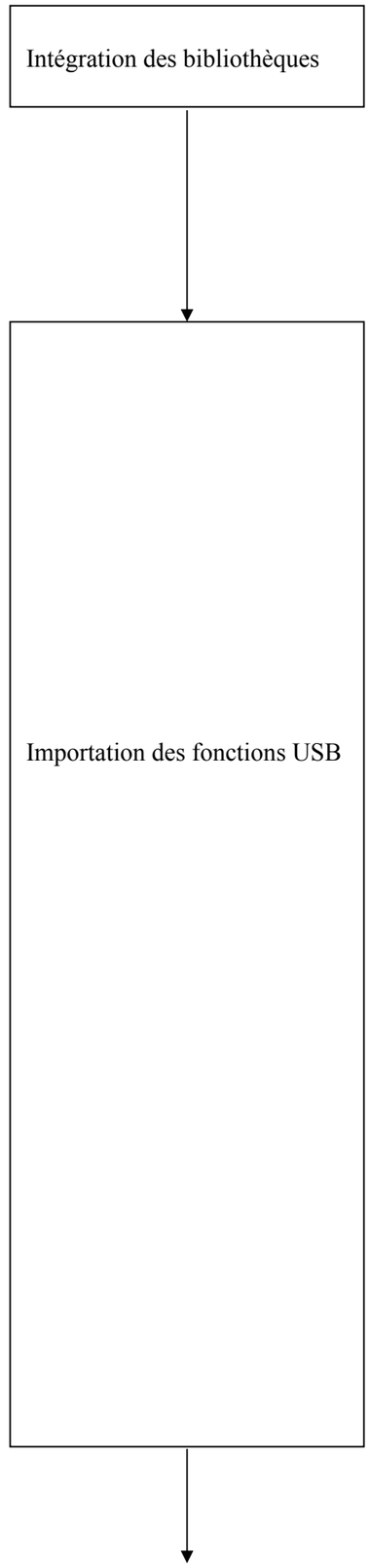
Étude et réalisation d'un ensemble caméra hyper spectrale

Dominique Lahaye

Annexe DA : Form1.cpp

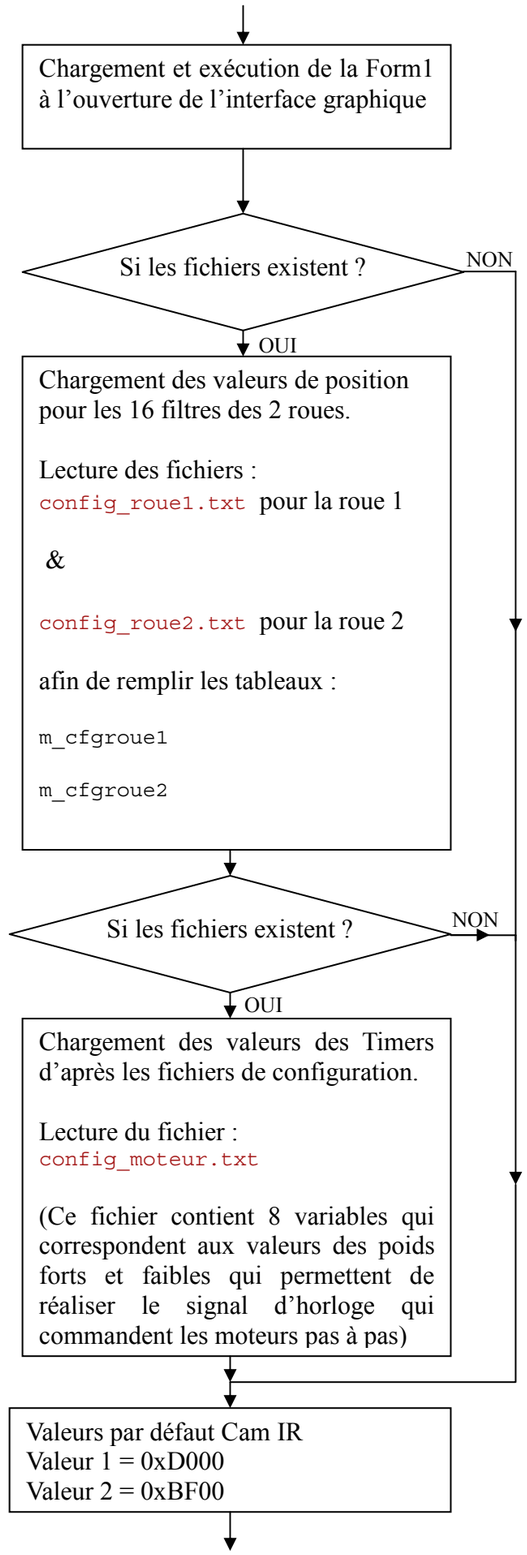
Annexe DA : Form1.cpp

```
#include "StdAfx.h"
#include "Form1.h"

namespace Cmd_pic {

	/********************************************* Importation fonction USB ***********************************************************/
	using namespace System::Threading;
	using namespace System::Runtime::InteropServices;  //Need this to support "unmanaged" code.

	#ifdef UNICODE
	#define  Seeifdef Unicode
	#else
	#define Seeifdef  Ansi
	#endif

	[DllImport("MPUSBAPI.dll" , EntryPoint="_MPUSBGetDLLVersion")]
	extern "C" DWORD MPUSBGetDLLVersion(void);
	[DllImport("MPUSBAPI.dll" , EntryPoint="_MPUSBGetDeviceCount")]
	extern "C" DWORD MPUSBGetDeviceCount(PCHAR pVID_PID);
	[DllImport("MPUSBAPI.dll" , EntryPoint="_MPUSBOpen")]
	extern "C" HANDLE MPUSBOpen(DWORD instance,        //  Input
	                            PCHAR pVID_PID,        // Input
	                            PCHAR pEP,             // Input
	                            DWORD dwDir, // Input
	                            DWORD dwReserved);// Input

	[DllImport("MPUSBAPI.dll" , EntryPoint="_MPUSBClose")]
	extern "C" BOOL MPUSBClose(HANDLE handle); //Input
	[DllImport("MPUSBAPI.dll" , EntryPoint="_MPUSBRead")]
	extern "C" DWORD MPUSBRead(HANDLE handle,          // Input
	                           PVOID pData,    // Output
	                           DWORD dwLen,// Input
	                           PDWORD pLength,        // Output
	                           DWORD dwMilliseconds);// Input

	[DllImport("MPUSBAPI.dll" , EntryPoint="_MPUSBWrite")]
	extern "C" DWORD MPUSBWrite(HANDLE handle,         // Input
	                            PVOID pData,    // Output
	                            DWORD dwLen,// Input
	                            PDWORD pLength,        // Output
	                            DWORD dwMilliseconds);// Input

	[DllImport("MPUSBAPI.dll" , EntryPoint="_MPUSBReadInt")]
	extern "C" DWORD MPUSBReadInt(HANDLE handle,       // Input
	                              PVOID pData,    // Output
	                              DWORD dwLen,// Input
	                              PDWORD pLength,        // Output
	                              DWORD dwMilliseconds);// Input

	[DllImport("user32.dll" , CharSet = CharSet::Seeifdef, EntryPoint="RegisterDeviceNotification")]
	extern "C" HDEVNOTIFY WINAPI RegisterDeviceNotificationUM(
	        HANDLE hRecipient,
	        LPVOID NotificationFilter,
	        DWORD Flags);

	//----------------Global variables used in this application-------------------------------
	HANDLE  EP1INHandle = INVALID_HANDLE_VALUE;
	HANDLE  EP1OUTHandle = INVALID_HANDLE_VALUE;
	BOOL AttachedState = FALSE;                //Need to keep track of the USB device attachment status for proper plug and play operation.
/*************************************************************************************************************************************/
```



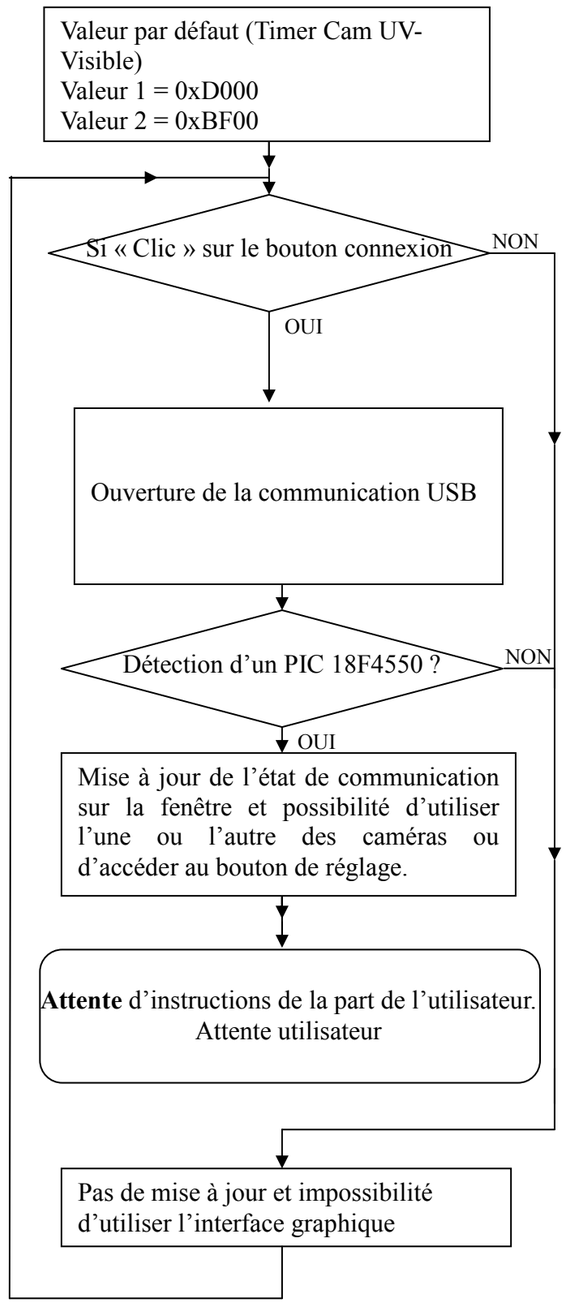
Annexe DA : Form1.cpp

```
System::Void Form1::Form1_Load(System::Object^  sender, System::EventArgs^  e)     //Fonction s'exécutant à l'ouverture de la fenêtre principale
{
	frm2 = gcnew Form2(this); //permet d'accéder au fonction de la form2
	groupBox_moteurs->Enabled = 0;  //groupbox CamIR grisé
	groupBox1->Enabled = 0;                    //groupbox CamUV grisé
	textBox_statutBox->Text = "Non connecté";
	if (File::Exists("config_roue1.txt") && File::Exists("config_roue2.txt")) //chargement de la configuration des servomoteurs
	{
		StreamReader ^sr1;
		String ^line1;
		StreamReader ^sr2;
		String ^line2;
		int i = 0;
		int j = 0;
		sr1 = gcnew StreamReader("config_roue1.txt");
		line1 = sr1->ReadLine();
		while (line1 != nullptr)
		{
		m_cfgroue1[i] = line1;
		line1 = sr1->ReadLine();
		i++;
		}
		sr2 = gcnew StreamReader("config_roue2.txt");
		line2 = sr2->ReadLine();
		while (line2 != nullptr)
		{
		m_cfgroue2[j] = line2;
		line2 = sr2->ReadLine();
		j++;
		}
		sr1->Close();
		sr2->Close();
	}
	if (File::Exists("config_moteur.txt"))          //chargement de la configuration des moteurs pas-à-pas
	{
		cli::array<String ^,1>^ tab_tmr = gcnew cli::array<String ^>(8); //création d'un tableau de string de 8 éléments
		StreamReader ^sr3;
		String ^line3;
		int k = 0;
		sr3 = gcnew StreamReader("config_moteur.txt");
		line3 = sr3->ReadLine();
		while (line3 != nullptr)
		{
			tab_tmr[k] = line3;
			line3 = sr3->ReadLine();
			k++;
		}
		sr3->Close();
		m_tmrH1 = Convert::ToInt32(tab_tmr[0], 16);
		m_tmrL1 = Convert::ToInt32(tab_tmr[1], 16);
		m_tmrH2 = Convert::ToInt32(tab_tmr[2], 16);
		m_tmrL2 = Convert::ToInt32(tab_tmr[3], 16);
		m_timerUVH1 = Convert::ToInt32(tab_tmr[4], 16);
		m_timerUVL1 = Convert::ToInt32(tab_tmr[5], 16);
		m_timerUVH2 = Convert::ToInt32(tab_tmr[6], 16);
		m_timerUVL2 = Convert::ToInt32(tab_tmr[7], 16);
}
else
		m_tmrH1 = 0xD0;
		m_tmrL1 = 0x00;
		m_tmrH2 = 0xBF;
		m_tmrL2 = 0x00;
```



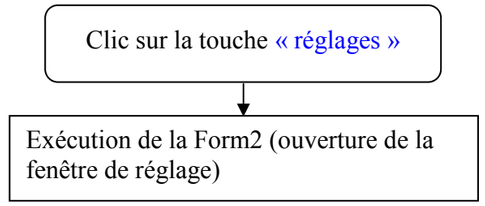
Annexe DA : Form1.cpp

```
                m_timerUVH1 = 0xD0;
                m_timerUVL1 = 0x00;
                m_timerUVH2 = 0xBF;
                m_timerUVL2 = 0x00;
}

System::Void Form1::button_connect_Click(System::Object^  sender, System::EventArgs^  e)     //Lors du clic sur le bouton "Connexion"
{
        /********************************************** Ouverture communication USB ************************************************************/
        //Globally Unique Identifier (GUID). Windows uses GUIDs to identify things.
        GUID InterfaceClassGuid = {0xa5dcbf10, 0x6530, 0x11d2, 0x90, 0x1F, 0x00, 0xC0, 0x4F, 0xB9, 0x51, 0xED}; //Globally Unique Identifier (GUID) for USB peripheral devices

        //Register for WM_DEVICECHANGE notifications:
        DEV_BROADCAST_DEVICEINTERFACE MyDeviceBroadcastHeader;// = new DEV_BROADCAST_HDR;
        MyDeviceBroadcastHeader.dbcc_devicetype = DBT_DEVTYP_DEVICEINTERFACE;
        MyDeviceBroadcastHeader.dbcc_size = sizeof(DEV_BROADCAST_DEVICEINTERFACE);
        MyDeviceBroadcastHeader.dbcc_reserved = 0;          //Reserved says not to use...
        MyDeviceBroadcastHeader.dbcc_classguid = InterfaceClassGuid;
        RegisterDeviceNotificationUM((HANDLE)this->Handle, &MyDeviceBroadcastHeader, DEVICE_NOTIFY_WINDOW_HANDLE);

        //Now perform an initial start up check of the device state (attached or not attached), since we would not have
        //received a WM_DEVICECHANGE notification.
        if(MPUSBGetDeviceCount(DeviceVID_PID))          //Check and make sure at least one device with matching VID/PID is attached
        {
                EP1OUTHandle = MPUSBOpen(0, DeviceVID_PID, "\\MCHP_EP1", MP_WRITE, 0);
                EP1INHandle = MPUSBOpen(0, DeviceVID_PID, "\\MCHP_EP1", MP_READ, 0);

                AttachedState = TRUE;
                textBox_statutBox->Text = "Connecté";
        }
        else        //Device must not be connected (or not programmed with correct firmware)
        {
                textBox_statutBox->Text = "Non connecté";
                AttachedState = FALSE;
        }
        /***************************************************************************************************************************************/
}
```

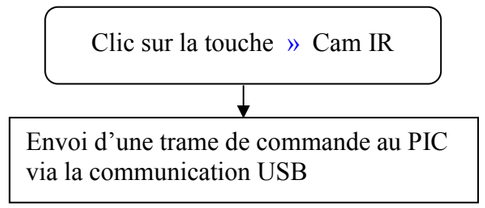


Annexe DA : Form1.cpp

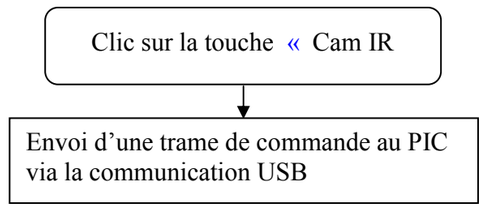
```
System::Void Form1::button_reglages_Click(System::Object^  sender, System::EventArgs^  e)    //Ouverture Form2 permettant les réglages
{
        frm2->ShowDialog();
}
```



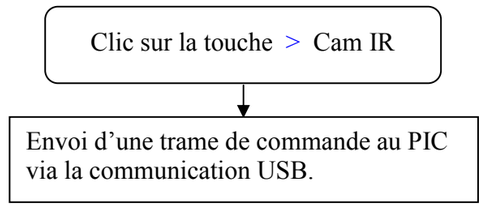

```
System::Void Form1::button_avfast_Click(System::Object^  sender, System::EventArgs^  e)              //Avance rapide tour par tour moteur caméra IR
{
        cmd_moteur_CamIR(m_tmrH1, m_tmrL1, 1, 1, 1, 1);
}
//                                                  Caméra IR
//                                                  Mode tour
//                                                  Validation du boîtier CS=1
//                                                  Sens (avance caméra)
//                                                  Poids Faible du Timer (valeur équivalente à vitesse rapide)
//                                                  Poids FORT du Timer (valeur équivalente à vitesse rapide)
```



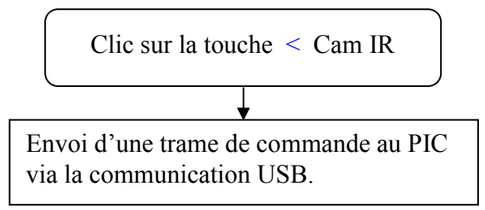

```
System::Void Form1::button_arfast_Click(System::Object^  sender, System::EventArgs^  e)              //Avance sens inverse rapide moteur caméra IR
{
        cmd_moteur_CamIR(m_tmrH1, m_tmrL1, 0, 1, 1, 1);
}

//                                                  Caméra IR
//                                                  Mode tour
//                                                  Validation du boîtier CS=1
/                                                   Sens (recul de la caméra)
//                                                  Poids Faible du Timer (valeur équivalente à vitesse rapide)
//                                                  Poids FORT du Timer (valeur équivalente à vitesse rapide)
```




```
System::Void Form1::button_avlow_Click(System::Object^  sender, System::EventArgs^  e)              //Avance lente moteur caméra IR
{
        cmd_moteur_CamIR(m_tmrH2, m_tmrL2, 1, 1, 1, 1);
}
//                                                  Caméra IR
//                                                  Mode tour
//                                                  Validation du boîtier CS=1
//                                                  Sens (avance caméra)
//                                                  Poids Faible du Timer (valeur équivalente à vitesse lente)
//                                                  Poids FORT du Timer (valeur équivalente à vitesse lente)
```




```
System::Void Form1::button_arlow_Click(System::Object^  sender, System::EventArgs^  e)              //Avance sens inverse lente moteur caméra IR
{
        cmd_moteur_CamIR(m_tmrH2, m_tmrL2, 0, 1, 1, 1);
}
//                                                  Caméra IR
//                                                  Mode tour
//                                                  Validation du boîtier CS=1
//                                                  Sens (recul de la caméra)
//                                                  Poids Faible du Timer (valeur équivalente à vitesse lente)
//                                                  Poids FORT du Timer (valeur équivalente à vitesse lente)
```



Annexe DA : Form1.cpp

```
System::Void Form1::button_avpas_Click(System::Object^  sender, System::EventArgs^  e)          //Avance pas-à-pas moteur caméra IR
{
        cmd_moteur_CamIR(m_tmrH2, m_tmrL2, 1, 1, 0, 1);
}
//                                                                  Caméra IR
//                                                                  Mode pas à pas
//                                                                  Validation du boîtier CS=1
//                                                                  Sens (avance caméra)
//                                                                  Poids Faible du Timer (valeur équivalente à vitesse lente)
//                                                                  Poids FORT du Timer (valeur équivalente à vitesse lente)
```

Clic sur la touche + Cam IR

Envoi d'une trame de commande au PIC via la communication USB.

```
System::Void Form1::button_arpas_Click(System::Object^  sender, System::EventArgs^  e)          //Avance sens inverse pas-à-pas moteur caméra IR
{
        cmd_moteur_CamIR(m_tmrH2, m_tmrL2, 0, 1, 0, 1);
}
//                                                                  Caméra IR
//                                                                  Mode pas à pas
//                                                                  Validation du boitier CS=1
//                                                                  Sens (recul de la caméra)
//                                                                  Poids Faible du Timer (valeur équivalente à vitesse lente)
//                                                                  Poids FORT du Timer (valeur équivalente à vitesse lente)
```

Clic sur la touche - Cam IR

Envoi d'une trame de commande au PIC via la communication USB

```
System::Void Form1::button_stop_Click(System::Object^  sender, System::EventArgs^  e)          //Arrêt moteur caméra IR
{
        cmd_moteur_CamIR(0, 0, 1, 0, 0, 1);
}
                                                                    Caméra IR
//                                                                  Mode pas à pas
//                                                                  Boitier non sélectionné CS=0 => ARRET du moteur
//                                                                  Sens (avance caméra)
//                                                                  Poids Faible du Timer (valeur équivalente à 0x00)
//                                                                  Poids FORT du Timer (valeur équivalente à 0x00)
```

Clic sur la touche Stop Cam IR

Envoi d'une trame de commande au PIC via la communication USB

```
System::Void Form1::button_arfastUV_Click(System::Object^  sender, System::EventArgs^  e)  //Descente rapide tour par tour moteur caméra UV
{
        cmd_moteur_CamUV(m_timerUVH1, m_timerUVL1, 0, 1, 1, 0);
}
                                                                    Caméra UV
//                                                                  Mode tours entiers
//                                                                  Validation du boitier CS=1
//                                                                  Sens (recul de la caméra)
//                                                                  Poids Faible du Timer (valeur équivalente à vitesse Rapide)
//                                                                  Poids FORT du Timer (valeur équivalente à vitesse Rapide)
```

Clic sur la touche « Cam UV

Envoi d'une trame de commande au PIC via la communication USB.

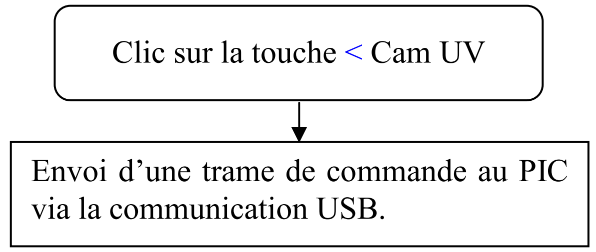
```
System::Void Form1::button_avfastUV_Click(System::Object^  sender, System::EventArgs^  e)  //Montée rapide tour par tour moteur caméra UV
{
        cmd_moteur_CamUV(m_timerUVH1, m_timerUVL1, 1, 1, 1, 0);
}
//                                                                  Caméra UV
//                                                                  Mode tours entiers
//                                                                  Validation du boitier CS=1
//                                                                  Sens (avance caméra)
//                                                                  Poids Faible du Timer (valeur équivalente à vitesse Rapide)
//                                                                  Poids FORT du Timer (valeur équivalente à vitesse Rapide)
```

Clic sur la touche » Cam UV

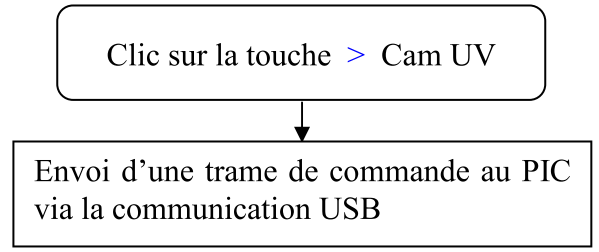
Envoi d'une trame de commande au PIC via la communication USB.

Annexe DA : Form1.cpp

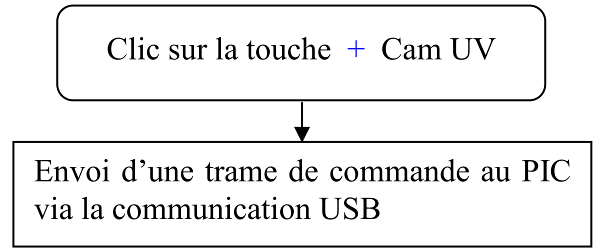
```
System::Void Form1::button_arlowUV_Click(System::Object^  sender, System::EventArgs^  e)  //Descente lente tour par tour moteur caméra UV
{
        cmd_moteur_CamUV(m_timerUVH2, m_timerUVL2, 0, 1, 1, 0);
}
//                                          Caméra UV
//                                          Mode tours entiers
//                                          Validation du boitier CS=1
//                                          Sens (recul de la caméra)
//                                          Poids Faible du Timer (valeur équivalente à vitesse lente)
//                                          Poids FORT du Timer (valeur équivalente à vitesse lente)
```



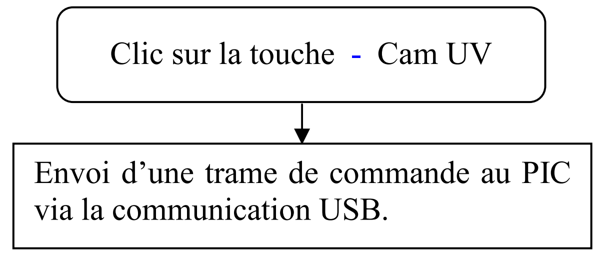

```
System::Void Form1::button_avlowUV_Click(System::Object^  sender, System::EventArgs^  e)  //Montée lente tour par tour moteur caméra UV
{
        cmd_moteur_CamUV(m_timerUVH2, m_timerUVL2, 1, 1, 1, 0);
}
//                                          Caméra UV
//                                          Mode tours entiers
//                                          Validation du boitier CS=1
//                                          Sens (avance de la caméra)
//                                          Poids Faible du Timer (valeur équivalente à vitesse lente)
//                                          Poids FORT du Timer (valeur équivalente à vitesse lente)
```



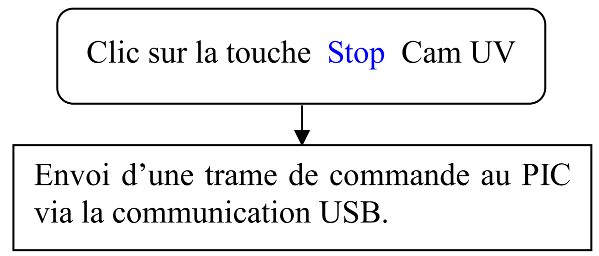

```
System::Void Form1::button_arpasUV_Click(System::Object^  sender, System::EventArgs^  e)  //Descente pas-à-pas moteur caméra UV
{
        cmd_moteur_CamUV(m_timerUVH2, m_timerUVL2, 0, 1, 0, 0);
}
//                                          Caméra UV
//                                          Mode pas à pas
//                                          Validation du boitier CS=1
//                                          Sens (recul de la caméra)
//                                          Poids Faible du Timer (valeur équivalente à vitesse lente)
//                                          Poids FORT du Timer (valeur équivalente à vitesse lente)
```




```
System::Void Form1::button_avpasUV_Click(System::Object^  sender, System::EventArgs^  e)  //Montée pas-à-pas moteur caméra UV
{
        cmd_moteur_CamUV(m_timerUVH2, m_timerUVL2, 1, 1, 0, 0);
}
//                                          Caméra UV
//                                          Mode pas à pas
//                                          Validation du boitier CS=1
//                                          Sens (avance de la caméra)
//                                          Poids Faible du Timer (valeur équivalente à vitesse lente)
//                                          Poids FORT du Timer (valeur équivalente à vitesse lente)
```

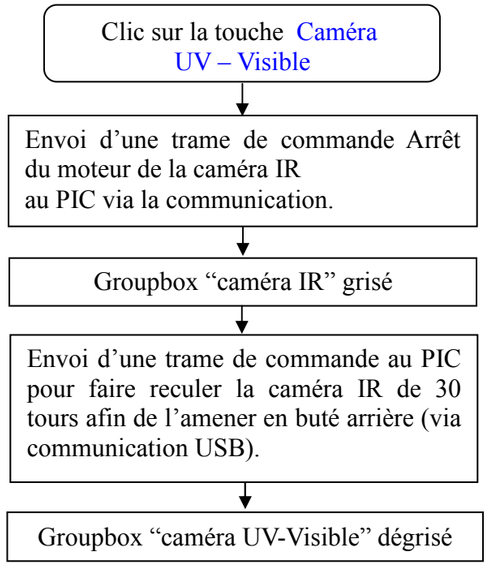



```
System::Void Form1::button_stopUV_Click(System::Object^  sender, System::EventArgs^  e)        //Arrêt moteur caméra UV
{
        cmd_moteur_CamUV(0, 0, 1, 0, 0, 0);
}
//                                          Caméra UV
//                                          Mode pas à pas
//                                          Boitier non sélectionné CS=0 => ARRET du moteur
//                                          Sens (avance caméra)
//                                          Poids Faible du Timer (valeur équivalente à 0x00)
//                                          Poids FORT du Timer (valeur équivalente à 0x00)
```

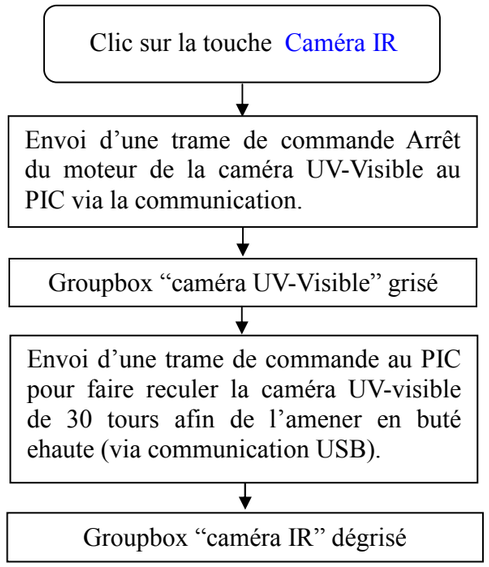


Annexe DA : Form1.cpp

```
System::Void Form1::camUV_Click(System::Object^  sender, System::EventArgs^  e)          //Lors du clic sur le bouton Caméra UV-Visible
{
        cmd_moteur_CamIR(0, 0, 1, 0, 0, 1);                                              //Arrêt déplacement caméra IR

//                                                                                       Caméra IR
//                                                                                       Mode pas à pas
//                                                                                       Boitier non sélectionné CS=0 => ARRET du moteur
//                                                                                       Sens (avance caméra)
//                                                                                       Poids Faible du Timer (valeur équivalente à 0x00)
//                                                                                       Poids FORT du Timer (valeur équivalente à 0x00)


        groupBox_moteurs->Enabled = 0;                                                   //groupbox caméra IR grisé


        int nb_tour = 30;
        textBox_nbtour->Text = nb_tour.ToString();
        cmd_moteur_CamIR(m_tmrH1, m_tmrL1, 1, 1, 1, 1);                                  //Recule jusqu'à fin de course arrière de la caméra IR

                                                                                         Caméra IR
//                                                                                       Mode tours entiers
//                                                                                       Validation du boitier CS=1
//                                                                                       Sens (avance caméra)
//                                                                                       Poids Faible du Timer (valeur équivalente à vitesse Rapide)
//                                                                                       Poids FORT du Timer (valeur équivalente à vitesse Rapide)

        groupBox1->Enabled = 1;                                                          //groupbox caméra UV-Visible dégrisé
}
```

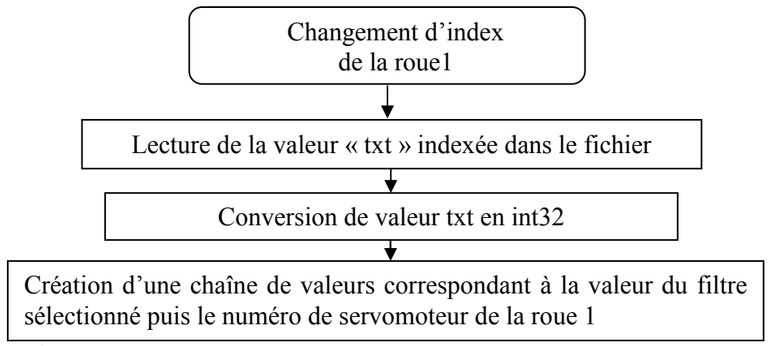



```
System::Void Form1::camIR_Click(System::Object^  sender, System::EventArgs^  e)          //Lors du clic sur le bouton Caméra IR
{
        cmd_moteur_CamUV(0, 0, 1, 0, 0, 0);                                              //Arrêt déplacement caméra UV-Visible

//                                                                                       Caméra UV
//                                                                                       Mode tours pas à pas
//                                                                                       Boitier non sélectionné CS=0 => ARRET du moteur
//                                                                                       Sens (avance caméra)
//                                                                                       Poids Faible du Timer (valeur équivalente à 0x00)
//                                                                                       Poids FORT du Timer (valeur équivalente à 0x00)

        groupBox1->Enabled = 0;                                                          //groupbox caméra UV-Visible grisé


        int nb_tour = 30;
        textBox_nbtourUV->Text = nb_tour.ToString();
        cmd_moteur_CamUV(m_timerUVH1, m_timerUVL1, 1, 1, 1, 0);                          //Montée jusqu'à fin de course haut de la caméra UV-Visible

//                                                                                       Caméra UV
//                                                                                       Mode tours entiers
//                                                                                       Validation du boitier CS=1
//                                                                                       Sens (avance caméra)
//                                                                                       Poids Faible du Timer (valeur équivalente à vitesse Rapide)
//                                                                                       Poids FORT du Timer (valeur équivalente à vitesse Rapide)

        groupBox_moteurs->Enabled = 1;                                                   //groupbox caméra IR dégrisé

}
```

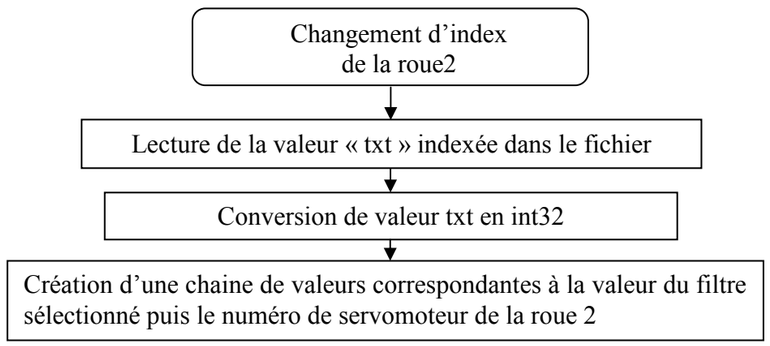


Annexe DA : Form1.cpp

```
System::Void Form1::comboBox_roue1_SelectedIndexChanged(System::Object^  sender, System::EventArgs^  e)          //Choix du filtre de la roue 1
        {
                String ^ txt = m_cfgroue1[comboBox_roue1->SelectedIndex];
                int val = Convert::ToInt32(txt);
                cmd_servo(val, 0);
        }
```

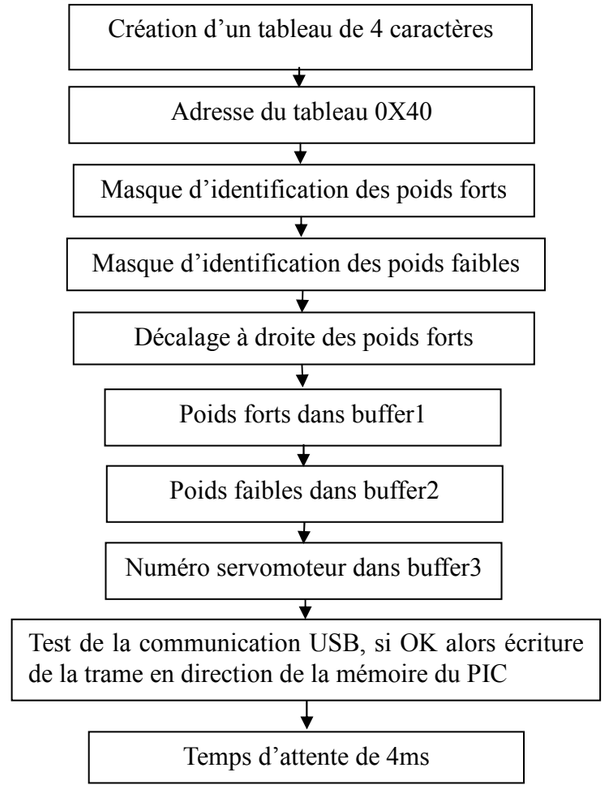



```
        System::Void Form1::comboBox_roue2_SelectedIndexChanged(System::Object^  sender, System::EventArgs^  e)          //Choix du filtre de la roue 2
        {
                String ^ txt = m_cfgroue2[comboBox_roue2->SelectedIndex];
                int val = Convert::ToInt32(txt);
                cmd_servo(val, 1);
        }
```



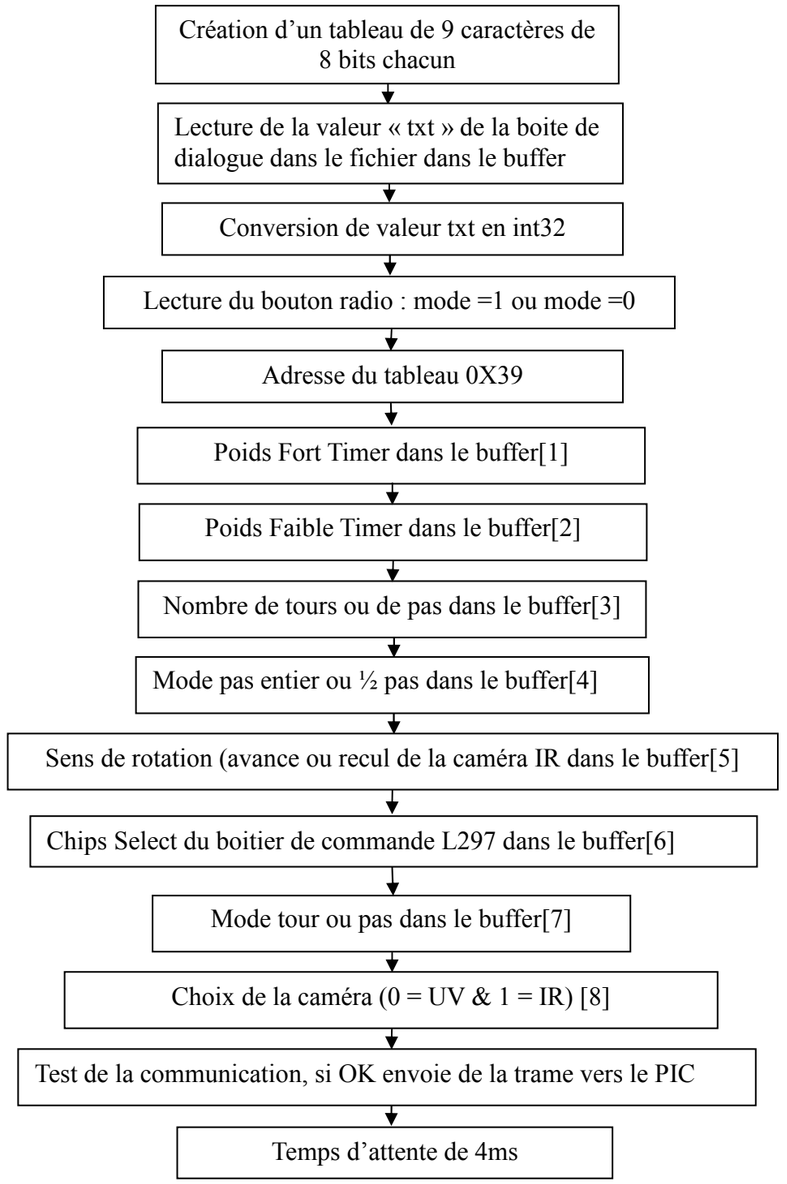

```
void Cmd_pic::Form1::cmd_servo(int val, int num_servo)          //Fonction d'envoi des paramètres des servomoteurs par l'USB          val: filtre choisi          num_servo: numéro de la roue(0 ou 1)
{
        unsigned char buffer[4];
        DWORD ActualLength;
        buffer[0] = 0x40;
        int msq1 = 0x00FF;
        int msq2 = 0xFF00;
        int result1 = val & msq1;
        int result2 = val & msq2;
        result2 = result2 >> 8;
        buffer[1] = result2;
        buffer[2] = result1;
        buffer[3] = num_servo;
        if(AttachedState == TRUE)
        {
                MPUSBWrite(EP1OUTHandle, buffer, 4, &ActualLength, 1000);          //Envoi des données au PIC par l'USB
        }
        Sleep(4);
}
```

Durée de la communication : 1 sec
Taille : 8 bits
Longueur des données : 4 éléments
Pointeur de donnée : Buffer



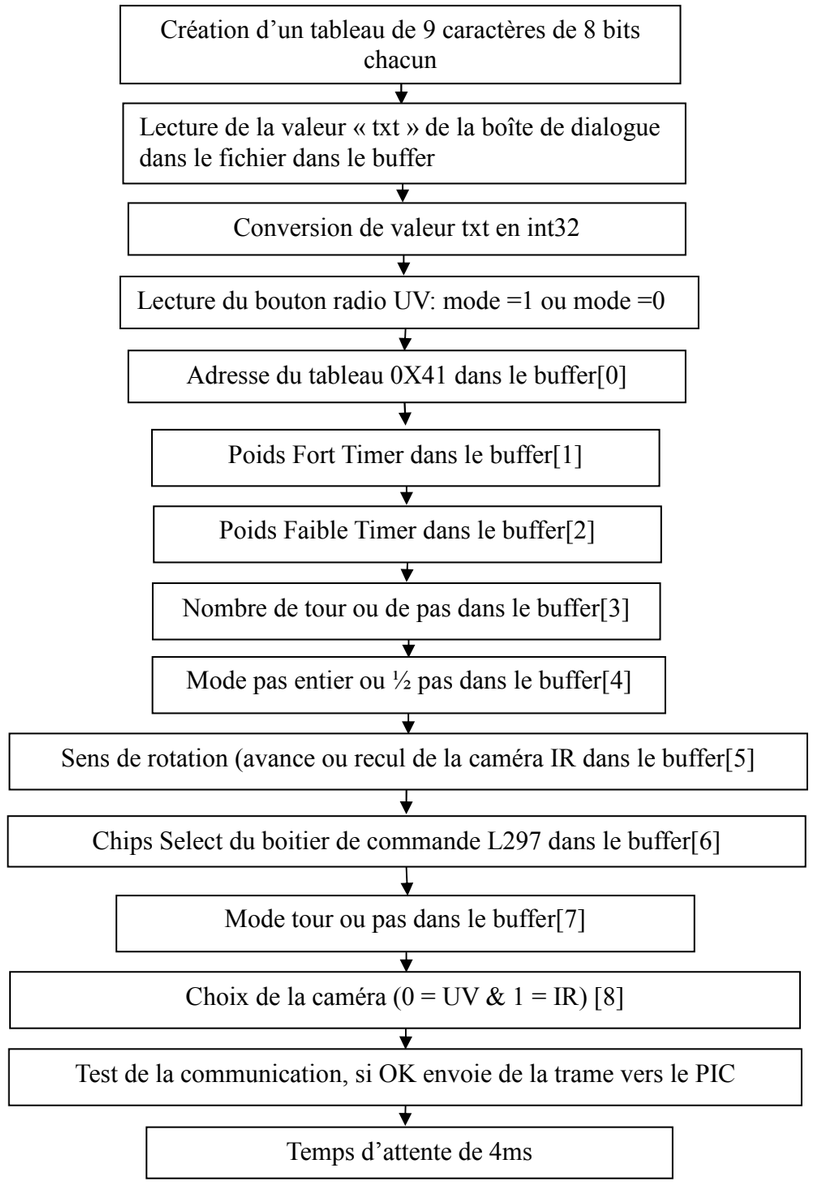
Annexe DA : Form1.cpp

```
void Cmd_pic::Form1::cmd_moteur_CamIR(int val_tmrH, int val_tmrL, bool sens, bool cs, bool mode_tour, bool choix_cam)                    //Fonction d'envoi des paramètres des moteurs pas-à-pas par l'USB
        {       //val_tmrH: valeur bits poids fort timer        val_tmrL: valeur bits poids faible timer        sens: sens rotation moteur cs: sélection boitier (activation moteur)        mode_tour: mode tour entier ou pas-à-pas    choix_cam: 0 cam UV/ 1 cam IR
                unsigned char buffer[9];
                DWORD ActualLength;
                String ^ txt = textBox_nbtour->Text;
                int nb_tour = Convert::ToInt32(txt);
                int mode;
                if (radioButton_pas->Checked) mode = 0;
                else mode = 1;
                buffer[0] = 0x39;
                buffer[1] = val_tmrH;                                   Poids Fort Timer dans le buffer[1]
                buffer[2] = val_tmrL;                                   Poids Faible Timer dans le buffer[1]
                buffer[3] = nb_tour;                                    Nombre de tour ou de pas
                buffer[4] = mode;                                       Mode pas entier ou ½ pas
                buffer[5] = sens;                                       Sens de rotation (avance ou recul de la caméra IR)
                buffer[6] = cs.                                         Chips Select du boitier de commande L297
                buffer[7] = mode_tour;                                  Mode tour entier ou pas à pas
                buffer[8] = choix_cam;                                  Choix de la caméra (0 = UV & 1 = IR)
                if(AttachedState == TRUE)
                {
                        MPUSBWrite(EP1OUTHandle, buffer, 9, &ActualLength, 1000);       //Envoi des données au PIC par l'USB
                }
                Sleep(4);
        }
```



Annexe DA : Form1.cpp

```
void Cmd_pic::Form1::cmd_moteur_CamUV(int val_tmrH, int val_tmrL, bool sens, bool cs, bool mode_tour, bool choix_cam)
	{
		unsigned char buffer[9];
		DWORD ActualLength;
		String ^ txt = textBox_nbtourUV->Text;
		int nb_tour = Convert::ToInt32(txt);
		int mode;
		if (radioButton_pasUV->Checked) mode = 0;
		else mode = 1;
		buffer[0] = 0x41;
		buffer[1] = val_tmrH;
		buffer[2] = val_tmrL;
		buffer[3] = nb_tour;
		buffer[4] = mode;
		buffer[5] = sens;
		buffer[6] = cs;
		buffer[7] = mode_tour;
		buffer[8] = choix_cam;
		if(AttachedState == TRUE)
		{
			MPUSBWrite(EP1OUTHandle, buffer, 9, &ActualLength, 1000);	//Envoi des données au PIC par l'USB
		}
		Sleep(4);
	}
```




```
void Cmd_pic::Form1::FermetureUSB(void)			//Fermeture de la communication USB
	{
	/********************************************** Fermeture communication USB ************************************************************/
	//Make sure to close any open handles, before exiting the application
	if (EP1OUTHandle != INVALID_HANDLE_VALUE)
		MPUSBClose (EP1OUTHandle);
	if (EP1INHandle != INVALID_HANDLE_VALUE)
		MPUSBClose (EP1INHandle);
	/***************************************************************************************************************************************/
	}
}
```