



Projet en ISN

IUT GEII MARSEILLE

Patrick GUMUCHIAN

Lycée Alphonse Benoit – L'Isle sur la Sorgue

Marc SILANUS

2011 - 2012

SOMMAIRE

1 - Objectifs.....	3
1.1 - Position du problème.....	3
1.2 - Mise en situation de l'élève.....	3
2 - Exemples de Projet : Contrôle/commande.....	4
2.1 - Cahier des charges.....	4
2.2 - Démarche à suivre.....	4
2.3 - Communication avec la carte d'acquisition.....	5
2.4 - Utilisation d'une librairie : "MSCOMM32.OCX".....	6
2.5 - Les contrôles de la librairie : "MSCOMM32.OCX".....	6
2.5.1 - Avec OpenOffice.....	6
2.5.2 - Avec Excel.....	7
2.6 - Pour la version 64 bits de Mscomm32 :.....	7
3 - Le Servomoteur.....	8
3.1 - Caractéristiques techniques : (sous 6V).....	8
3.2 - Principe de fonctionnement :.....	8
3.3 - Connexion Carte Picdem2 Plus – servomoteur.....	9
3.4 - Schéma structurel de la carte Servomoteur.....	9
4 - Le ventilateur.....	10
4.1 - Description.....	10
4.2 - Caractéristiques constructeur.....	10
4.3 - Fonctionnement.....	10
4.4 - Schéma structurel Carte Ventilateur.....	11
4.5 - Implantation des composants de la carte Ventilateur.....	11
4.6 - Connexion Carte Pucdem2 Plus – Ventilateur.....	12
5 - Travail à réaliser.....	12
6 - Exemples de Projet : Communication sériele.....	13
6.1 - Cahier des charges.....	13
6.2 - Démarche à suivre.....	13
7 - Travail à réaliser.....	14
8 - Exemples de Projet : Traitement d'image.....	15
8.1 - Cahier des charges.....	15
8.2 - Démarche à suivre.....	15
9 - Travail à réaliser.....	15
10 - Annexes.....	16
10.1 - Le composant SerialPort.....	16
10.1.1 - Principales propriétés.....	16
10.1.2 - Principales méthodes.....	16
10.1.3 - Principal événement.....	16
10.2 - Le composant OpenFileDialog.....	17
10.2.1 - Principales propriétés.....	17
10.2.2 - Principale méthode.....	17
10.3 - La classe Graphics.....	17
10.3.1 - Espace de nom.....	17
10.3.2 - Définition.....	17
10.3.3 - Principales méthodes.....	17
10.4 - La classe Bitmap.....	18
10.4.1 - Espace de nom.....	18
10.4.2 - Définition.....	18
10.4.3 - Constructeur.....	19
10.4.4 - Principales propriétés.....	19
10.4.5 - Principales méthodes.....	19

1 - Objectifs

1.1 - Position du problème

- ☞ Réaliser une application (logicielle et / ou matérielle) concrète à partir d'un cahier des charges avec un maximum d'autonomie.
- ☞ Des documents ressources seront mis à la disposition des élèves. Ces documents doivent fournir une aide pour trouver la solution au problème.

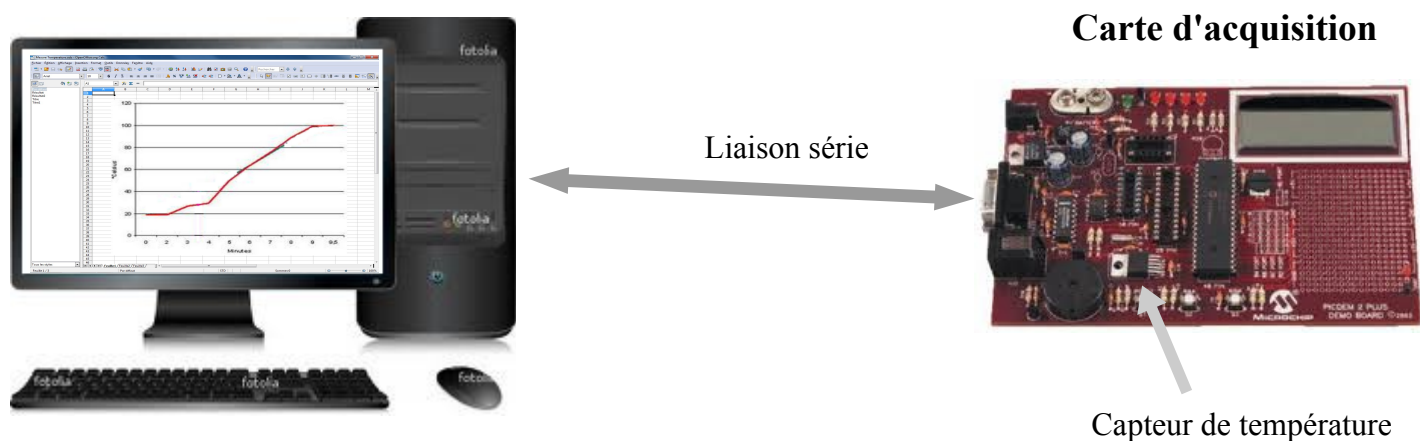
1.2 - Mise en situation de l'élève

- ☞ Faire une démarche de réflexion pour **analyser** le problème :
 - Quelles connaissances théoriques manquent ils ?
- ☞ Faire une démarche de réflexion pour **trouver** une ou plusieurs solutions au problème :
 - Quels matériels, applications et logiciels peut on utiliser ?
- ☞ Faire une démarche de réflexion pour **choisir** une solution au problème :
 - Evaluer la difficulté pour réaliser la solution et prendre en compte aussi le critère économique.

2 - Exemples de Projet : Contrôle/commande

2.1 - Cahier des charges

- ☞ Acquisition de température par liaison série dans le tableur d'OpenOffice pour une étude des variations de température dans une salle de classe pendant 24 h.
- ☞ Piloter un servomoteur par liaison série
- ☞ Piloter un ventilateur par liaison série



2.2 - Démarche à suivre

- ☞ Les connaissances théoriques :
 - Programmer un tableur : Basic, Python, JavaScript...
 - Protocole de communication RS232.
 - Partie logicielle permettant de communiquer avec la liaison série.
 - Programmation du microcontrôleur ou utilisation d'une carte déjà programmée.
- ☞ Les outils et le matériel à disposition dans le labo :
 - Les tableurs à disposition (Excel, OpenOffice).
 - Carte d'acquisition disponible : à faire ou à acheter.
 - Les langages de programmation : C, Basic, Python ...
- ☞ Choisir une solution réaliste avec le matériel et le logiciel disponible dans le labo :
 - Logiciel : OpenOffice avec programmation en Basic.
 - Matériel : carte Picdem2 Plus (Microchip) déjà programmée avec protocole de communication connue.

2.3 - Communication avec la carte d'acquisition

Pour communiquer il faut configurer le port série en :

- ☞ 9600 bauds
- ☞ 8 bits de données
- ☞ 1 bit de Stop
- ☞ Pas de parité

Le protocole de communication est le suivant :

- ☞ Allumer ou éteindre la led sur RB3 : envoyer le caractère "**w**"
- ☞ Allumer ou éteindre la led sur RB2 : envoyer le caractère "**x**"
- ☞ Pour stopper l'acquisition de température : envoyer les caractères "**az**"
- ☞ Pour reprendre l'acquisition de température : envoyer les caractères "**rz**"
- ☞ Changer la durée d'acquisition de température : envoyer la série de caractère "**h---z**" où les --- représente la **durée en seconde**. Pour envoyer 2 s, il suffit de taper "h2z".
- ☞ La carte envoie en permanence vers le PC, au rythme de la durée d'acquisition, le message suivant :

HxxxxxTyy + le code ASCII 10

avec **xxxxxx** représente le temps indiqué sur l'afficheur LCD de la carte et **yy** la température en degré Celsius.

- ☞ Changer l'angle du Servomoteur : envoyer la série de caractère "**s---z**" où les --- représente l'angle en **degré compris entre 0 et 200**. Pour envoyer 20°, il suffit de taper "s20z".
- ☞ Changer la vitesse du ventilateur : envoyer la série de caractère "**v---z**" où les --- représente le **pourcentage de la vitesse maxi**, valeur comprise **entre 0 et 100**. Pour envoyer 30%, il suffit de taper "v30z".
- ☞ Pour réinitialiser la communication : envoyer le caractère "**ESC**" (Touche **ESC** sur le clavier dont le code ASCII est **27** en décimal ou **1b** en hexa).

2.4 - Utilisation d'une librairie : "MSCOMM32.OCX"

Ce fichier ne fonctionne qu'avec un environnement 32 bits !!!!!

Si le fichier "MSCOMM32.OCX" n'est pas présent dans le dossier :

"c:\windows\system32\mscomm32.ocx"

alors il faut l'installer en suivant la procédure suivante :

Copier le fichier "MSCOMM32.OCX" dans le dossier :

c:\windows\system32\

Puis utiliser l'application "regsvr32" de Microcoft :

Aller sur le menu Démarrer puis Exécuter

et taper le ligne de commande suivante : regsvr32 mscomm32.ocx

C'est fait.

Il se peut que le fichier MSCOMM32.OCX ne soit pas utilisable dans Excel à cause de la base de registre verrouillée.

Il faut pour cela exécuter le fichier : "Mscomm_Pour_Excel.reg" qui modifie la base de registres.

Contenu du fichier "Mscomm_Pour_Excel.reg" :

Windows Registry Editor Version 5.00

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\
{648A5600-2C6E-101B-82B6-000000000014}]
```

```
"Compatibility Flags"=dword:00000000
```

2.5 - Les contrôles de la librairie : "MSCOMM32.OCX"

2.5.1 - Avec OpenOffice

Dim MSComm1 as object	' Declaration d'une variable objet
MSComm1 = CreateObject ("MSCOMMLib.MSComm")	' Creation de l'objet utilisant Mscomm32.ocx
MSComm1.Commport = 4	' Numero du port série
MSComm1.Settings = "9600,N,8,1"	' Protocole de communication
MSComm1.InputLen = 0	' Lecture complete du buffer de reception
MSComm1.InputMode = 0	' 0 = Mode texte - 1 = Mode Binaire
MSComm1.PortOpen = True	' Ouverture du port série
Entree = MSComm1.input	' Lecture du buffer de reception
MSComm1.output = Sortie	' Ecriture sur le port série
MSComm1.PortOpen = false	' Fermeture du port série

Voir le fichier : "Gestion Port Serie.ods"

2.5.2 - Avec Excel

Il faut utiliser un userform et ajouter un contrôle supplémentaire : "MSCOMM32.OCX"

2.6 - Pour la version 64 bits de Mscomm32 :

Il existe un contrôle nommé XMCOMMCRC.OCX

Seule différence avec MSCOMM32, c'est la propriété **INPUT** qui s'appelle **INPUTDATA**.

Installer le setup.exe du controle XMComm téléchargeable à cette adresse :
<http://home.comcast.net/~hardandsoftware/XMCommMin.zip>

Ensuite, la réponse est valable pour office 2007 mais je suppose que le principe est le même pour Excel 2010.

- ☞ Dans Excel, aller dans microsoft visual basic (alt + F11).
- ☞ Activer la boîte à outils où il y a les contrôles classiques (textbox, checkbox,...).
- ☞ Faire un clic droit dans une zone vide de la boîte à outils et sélectionner contrôles supplémentaires.
- ☞ Normalement, le contrôle s'appelle "XMComCRCPorts.XMCommCRC". Le sélectionner et miracle.

A vérifier !!!!!

3 - Le Servomoteur

Le HS-322HD est un servomoteur standard.

3.1 - Caractéristiques techniques : (sous 6V)

- ☞ Couple max : 3,5Kg.cm
- ☞ Courant max : 800 mA
- ☞
- ☞ Vitesse : 0,15sec / 60°
- ☞ Dimensions : 41x20x37mm
- ☞ Poids : 43g
- ☞ Pignons : nylon extra dur
- ☞ Prises : Hitec/JR - Futaba



Pulse Data :

All Hitec servos require 3-5V peak to peak square wave pulse.
Pulse duration is from 0.9ms to 2.1ms with 1.5ms as center.
The pulse refreshes at 50Hz (20ms).

Voltage Range :

All Hitec Servos can be operated within a 4.8V-6V. range.

Wire Color Meanings :

On all Hitec servos

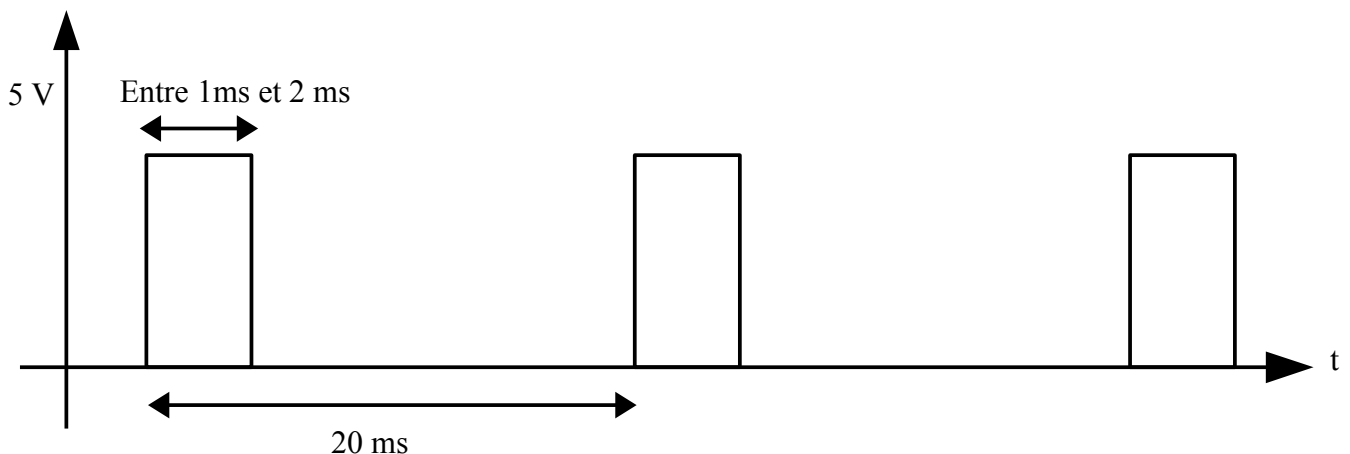
- ☞ the Black wire is 'ground',
- ☞ the Red wire (center) is 'power'
- ☞ hird wire is 'signal'.

Direction of Rotation :

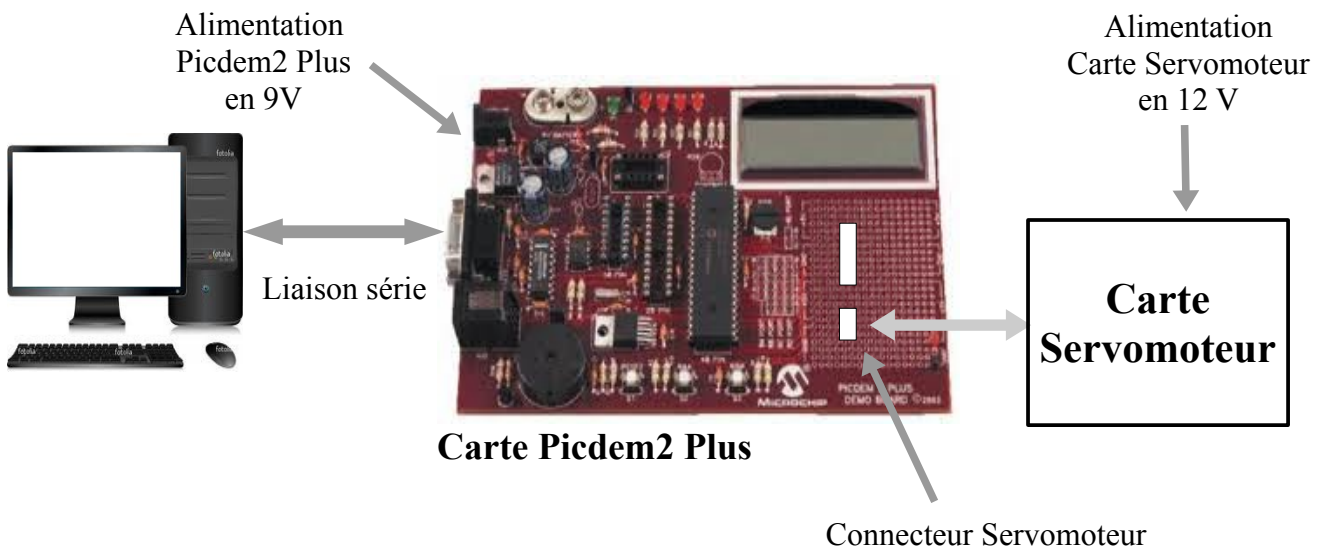
All Hitec servos turn Clockwise direction (CW).

3.2 - Principe de fonctionnement :

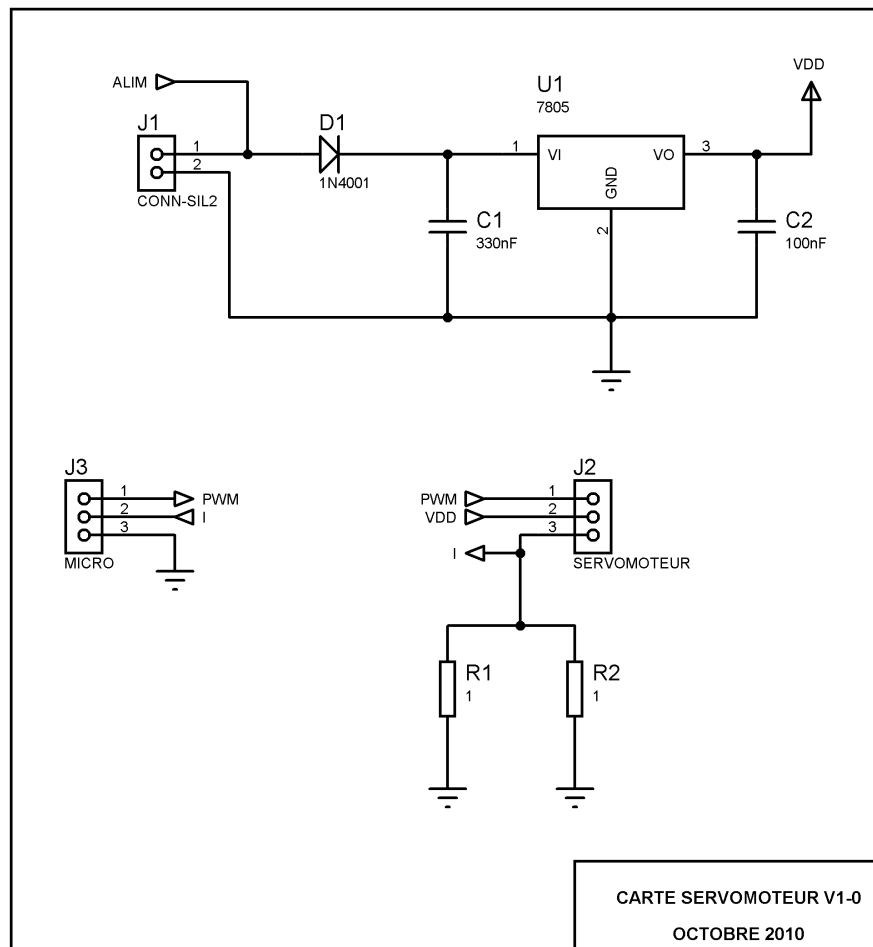
Fil de signal sur RC1 du microcontrôleur



3.3 - Connexion Carte Picdem2 Plus – servomoteur



3.4 - Schéma structurel de la carte Servomoteur



4 - Le ventilateur

4.1 - Description

Le ventilateur est constitué d'un moteur Brushless 2 paires de pôles (4 pôles).

4.2 - Caractéristiques constructeur

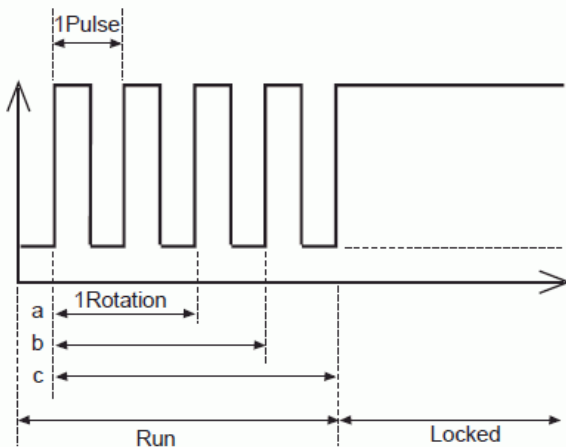
☞ Fan size	: 60x60x20mm
☞ Noise level	: 19.99 dBA
☞ Airflow	: 12.31 CFM (Cubic Feet per Minute)
☞ Fan speed	: 2500 RPM
☞ Connector	: 3-Pin Molex
☞ Operating Voltage	: 7 Volt - 12 Volt DC
☞ Input Power	: 0.08 A
☞ Lifetime	: 50.000 hours



Unité de mesure anglaise :

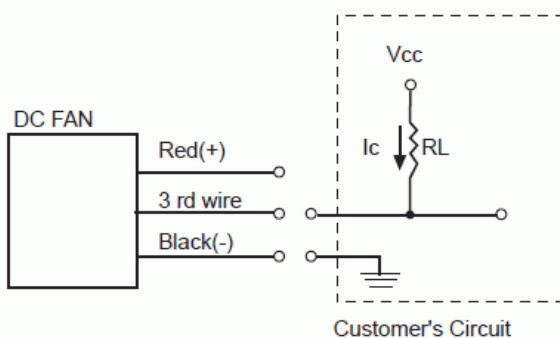
- ☞ 1 inch = 2,54 cm
- ☞ 1 foot = 12 inches = 30,48 cm

4.3 - Fonctionnement

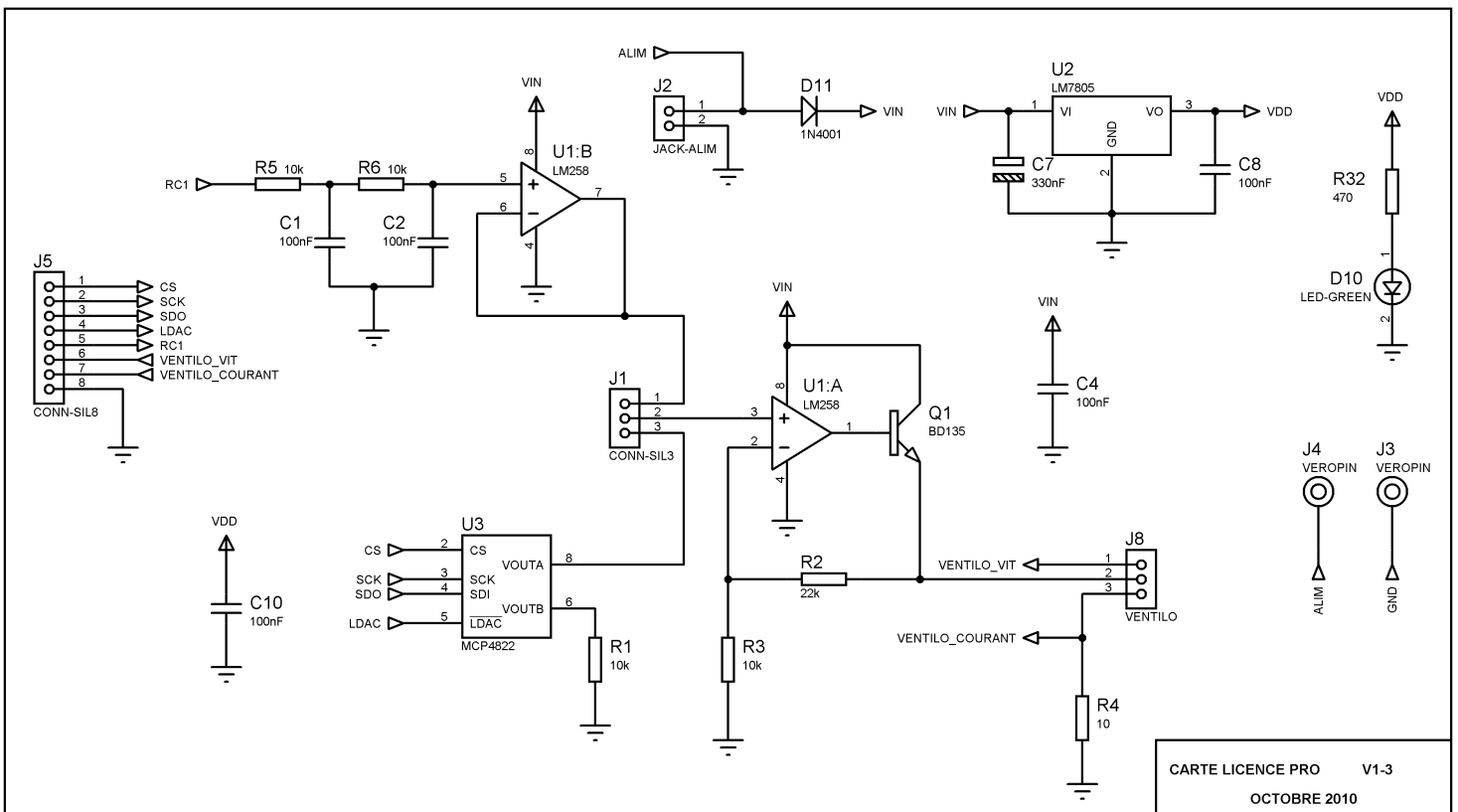


The relationship between rotation & output pulses signal from 3rd wire are as follows:

- (a) 1 Rotation=2 Pulses(4 poles' motor)
- (b) 1 Rotation=3 Pulses(6 poles' motor)
- (c) 1 Rotation=4 Pulses(8 poles' motor)

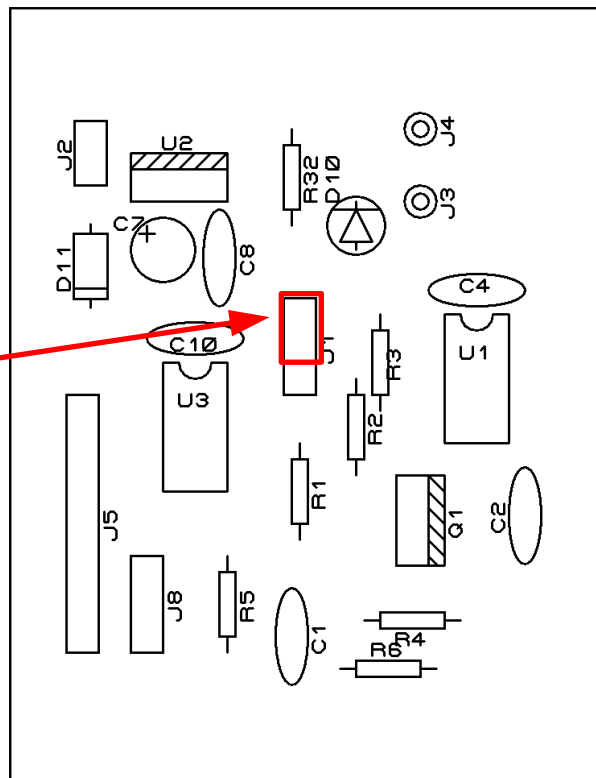


4.4 - Schéma structurel Carte Ventilateur

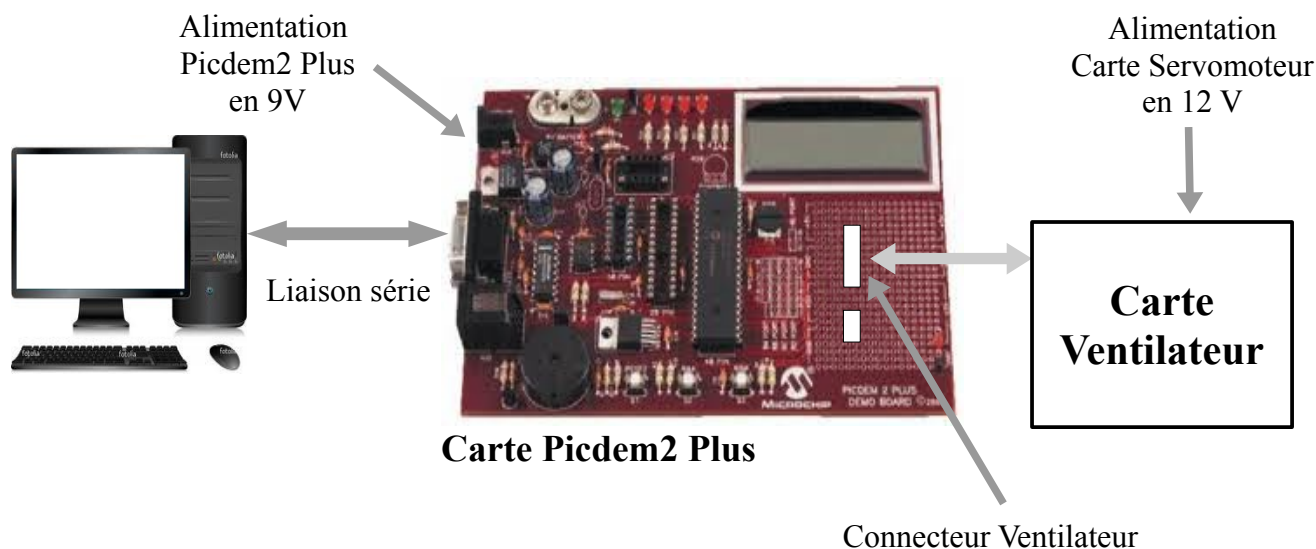


4.5 - Implantation des composants de la carte Ventilateur

Placer le cavalier J1 dans la position haute, comme indiqué sur le schéma



4.6 - Connexion Carte Picdem2 Plus – Ventilateur



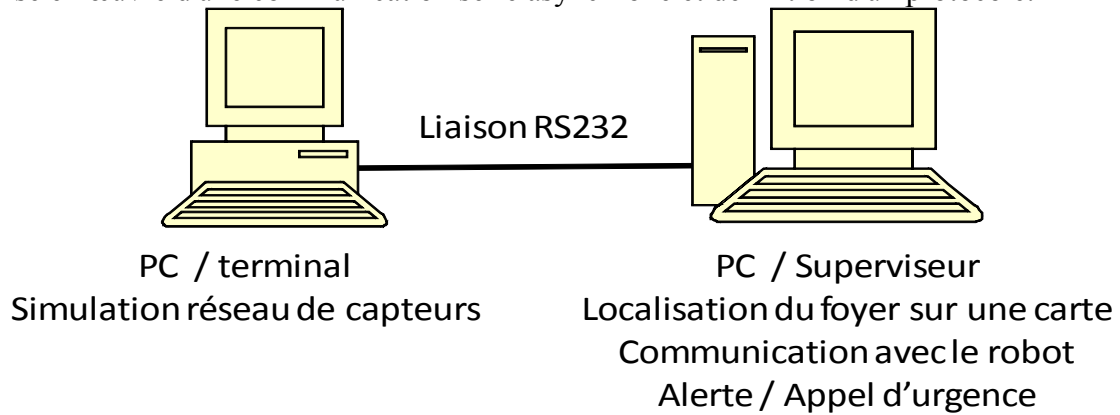
5 - Travail à réaliser

- ☞ Réaliser les câblages entre le PC, la carte Picdem2 Plus, la carte Servomoteur et la carte Ventilateur. **Les câblages se font hors tension.**
- ☞ Faire vérifier avant d'alimenter les différentes cartes.
- ☞ Ouvrir un hyperterminal sous Windows afin de vérifier le bon fonctionnement du système.
- ☞ Choisir un outil de développement (Visual C ou tableur) et remplacer l'hyperterminal par votre application.
 - Prévoir une démarche de développement et avancer par étape :
 - ⇒ étape 1 : utilisation des entrées sorties de l'outil de développement (par exemple lire ou écrire dans une cellule du tableur).
 - ⇒ étape 2 : mise en place de la communication série en émission
 - ⇒ étape 3 : mise en place de la communication série en réception
 - ⇒ étape 4 : faire les algorithmes
 - ⇒ étape 5 : réaliser les programmes et valider au fur et à mesure
- ☞ Appeler un formateur si vous souhaitez visualiser quelques signaux intéressants comme le signal de la liaison série RS232, le PWM du servomoteur ou celui du ventilateur.

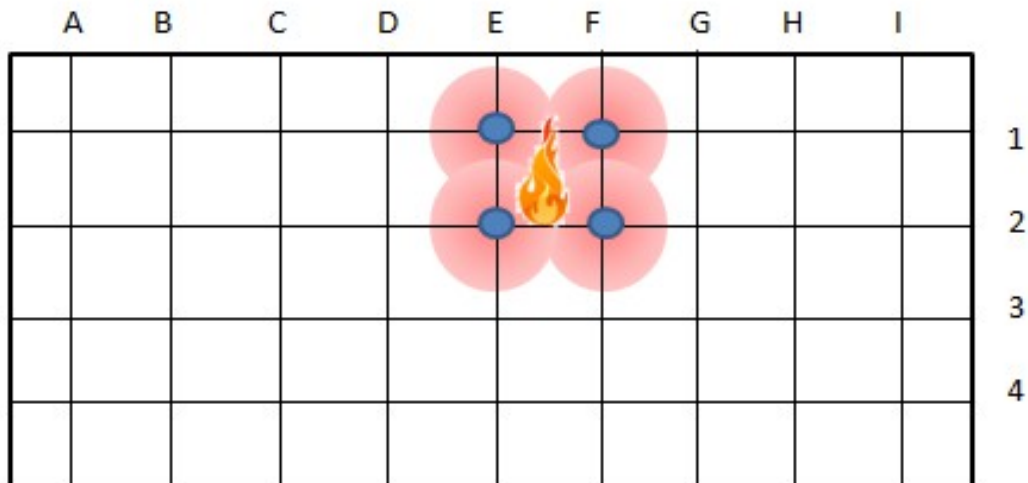
6 - Exemples de Projet : Communication sériele

6.1 - Cahier des charges

- ☞ Fournir une application de supervision permettant la localisation d'un incendie.
- ☞ Mise en œuvre d'une communication série asynchrone et définition d'un protocole.



Zone à surveiller



- ☞ Transmission des coordonnées des capteurs qui ont détecté un départ de feu : E1;F1;E2;F2;
- ☞ Protocole : transmission de 3 caractères **Colonne Ligne** ;

6.2 - Démarche à suivre

- ☞ Les connaissances théoriques :

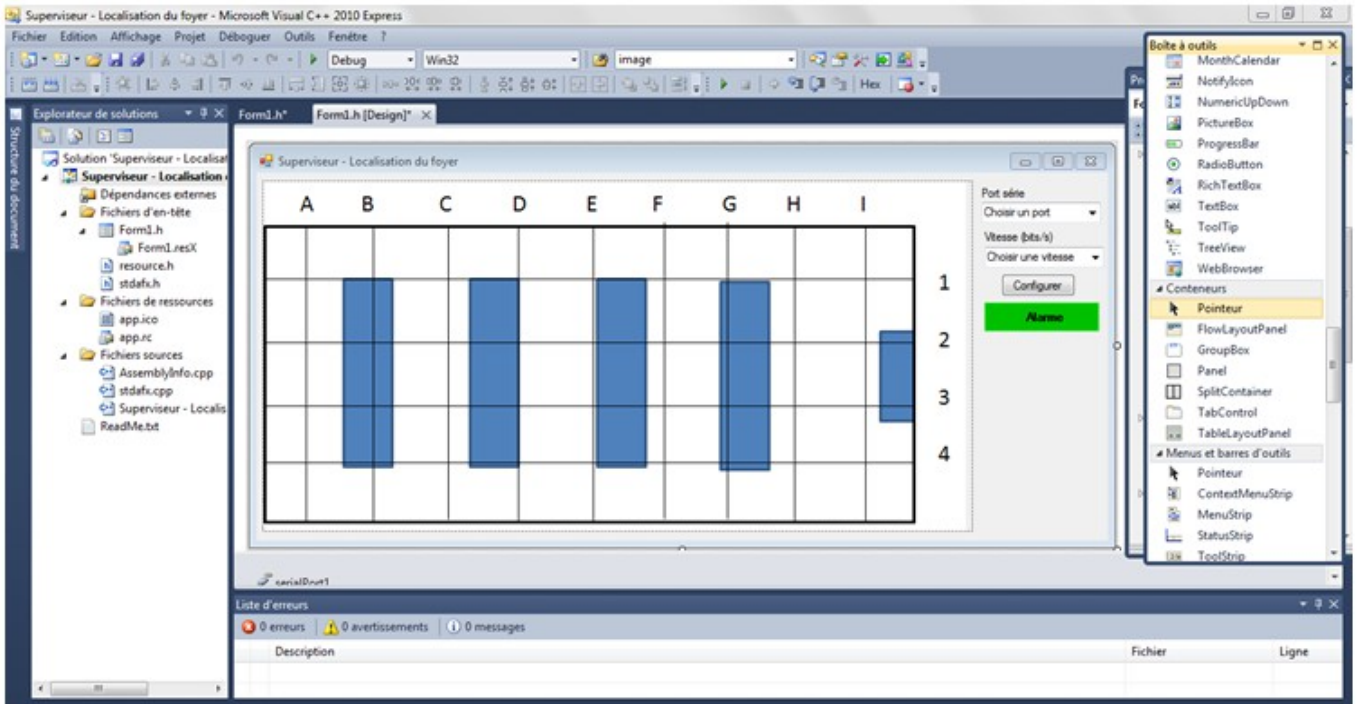
- Programmation Basic, C++, C#, Python, Java...
- Protocole de communication RS232.

- ☞ Les outils et le matériel nécessaire :

- Un logiciel de développement rapide (IDE : Integreted Developpement Environement) : Delphi, BuilderC++, Visual Studio, Qt Creator,
- 2 PC avec un logiciel de terminal : Hyperterminal, Putty, Minicom, terminal, ...

7 - Travail à réaliser

- ☞ Relier 2 PC entre-eux en utilisant un câble série. Si les PC ne disposent pas de ports séries, utiliser un adaptateur USB/Série.
- ☞ Ouvrir un terminal sur les PC, configurer la connexion selon votre choix et tester le transfert de données.
- ☞ Choisir un outil de développement et remplacer un terminal par votre application de supervision.



- ☞ Appeler un formateur si vous souhaitez visualiser le signal de la liaison série RS232.

Remarque : Ce projet peut être adapté pour réaliser une application de jeu comme le morpion ou la bataille navale. On devra alors réaliser une application permettant de gérer l'émission et la réception de données.

8 - Exemples de Projet : Traitement d'image

8.1 - Cahier des charges

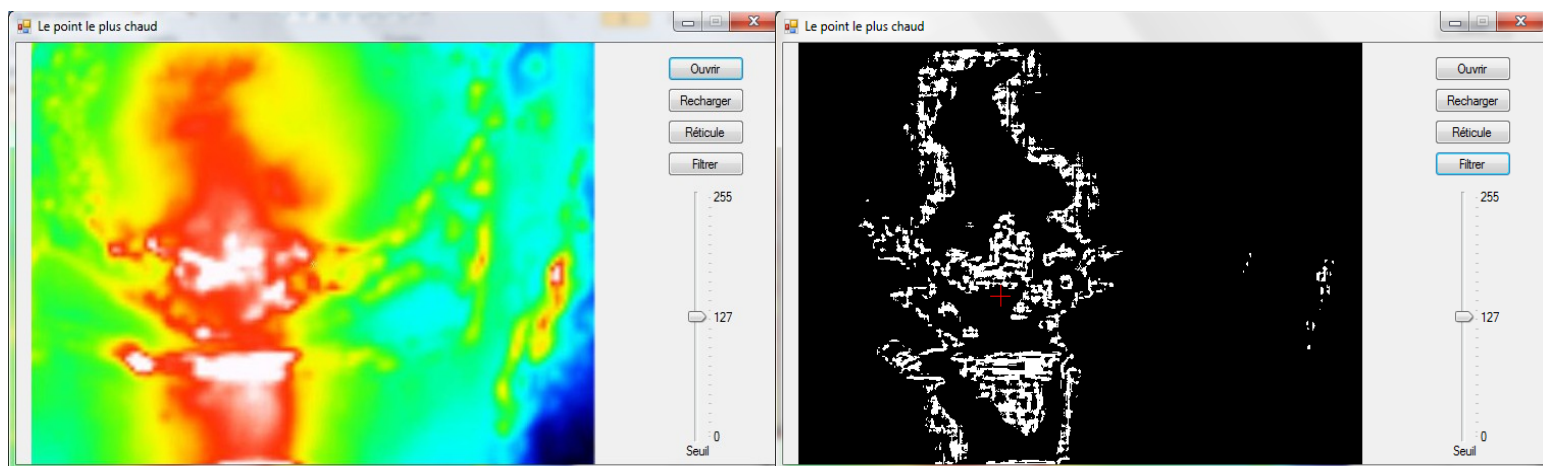
- ☞ Fournir une application de traitement d'image permettant d'identifier le centre du foyer d'un incendie.
- ☞ L'application doit permettre de :
 - Charger une image issue d'une caméra thermique.
 - Régler le seuil de sensibilité de température.
 - Matérialiser le centre du foyer (calcul du barycentre).

8.2 - Démarche à suivre

- ☞ Les connaissances théoriques :
 - Constitution d'une image BMP.
 - Algorithme de calcul du barycentre.
- ☞ Les outils et le matériel nécessaire :
 - Un logiciel de développement rapide (IDE : Integreted Developpement Environement) : Delphi, BuilderC++, Visual Studio, Qt Creator,
 - Une caméra thermique ou à défaut, plusieurs images thermiques stockées sur le PC.

9 - Travail à réaliser

- ☞ Ouvrir une image thermique dans Paint et identifier les valeurs des couleurs représentatives d'un point chaud.
- ☞ Concevoir l'application permettant de charger une image à l'écran.
- ☞ Ajouter un bouton pour filtrer les couleurs Rouge, Vert et Bleu selon l'algorithme suivant :
 - Si ComposanteRouge = 255 et si ComposanteVerte > 127 alors CouleurPixel = Blanc
 - Sinon CouleurPixel = Noir.
- ☞ Ajouter un curseur pour régler la valeur du seuil de sensibilité de la température (proportion de vert dans l'image) et modifier l'algorithme précédant pour tenir compte de se réglage.
- ☞ Calculer et afficher une petite croix au centre du foyer.



10 - Annexes

Les composants et classes suivants sont issues de Visual Studio Express. Les exemple de code sont en C++ mais peuvent facilement être traduit dans d'autres langages.

10.1 - Le composant SerialPort

Représente une ressource de port série.

10.1.1 - Principales propriétés

Name : Nom de l'objet instancié.

BaudRate : Obtient ou définit la vitesse en bauds série.

DataBits : Obtient ou définit la longueur standard des bits de données par octet.

Parity : Obtient ou définit le protocole de contrôle de parité.

StopBits : Obtient ou définit le nombre standard de bits d'arrêt par octet.

Handshake : Obtient ou définit le type de contrôle de flux pour la transmission de données par le port série.

PortName : Obtient ou définit le port pour les communications, y compris, de manière non limitative, tous les ports COM disponibles.

10.1.2 - Principales méthodes

Open () : Ouvre une nouvelle connexion au port série.

Close () : Ferme la connexion au port série.

ReadTo (String ^ value) : Lit une chaîne jusqu'au value spécifié. Retourne une chaîne de caractères.

ReadExisting () : Lit tous les octets immédiatement disponibles, en fonction de l'encodage, dans le flux et la mémoire tampon d'entrée de l'objet SerialPort. Retourne une chaîne de caractères.

Write (String^ text) : Écrit la chaîne spécifiée au port série.

10.1.3 - Principal événement

DataReceived : Représente la méthode qui générera l'événement reçu avec les données d'un objet SerialPort.

ex :

```
private: System::Void sp_DataReceived(System::Object^ sender,
System::IO::Ports::SerialDataReceivedEventArgs^ e)
{
    String^ indata = sp->ReadExisting();
    Label->Text = "Données reçues :" + indata;
}
```


10.2 - Le composant OpenFileDialog

Permet d'afficher une boîte de dialogue qui invite l'utilisateur à ouvrir un fichier.

10.2.1 - Principales propriétés

Name : Nom de l'objet instancié.

Filter : Définit la chaîne de filtrage des noms de fichiers en cours

ex: **Text Files (*.txt)|*.txt**

DefaultExt : Définit l'extension de nom de fichier par défaut.

ex: **txt**

FileName : Obtient ou définit une chaîne (de type **String**) comportant le nom de fichier sélectionné dans la boîte de dialogue d'ouverture de fichier.

10.2.2 - Principale méthode

ShowDialog () : Exécute une boîte de dialogue standard d'ouverture de fichier.

Retourne le choix de l'utilisateur (OK ou Annuler). La valeur de retour est de type **System.Windows.Forms.DialogResult** et les valeurs possibles sont :

- **Cancel**
- **OK**

```
ex :
if(openFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)
{
    //Do something
}
```

10.3 - La classe Graphics

Encapsule une surface de dessin GDI+ (Graphic Device Interface). Nécessaire pour afficher une image dans la fenêtre de l'application.

10.3.1 - Espace de nom

```
Using namespace System::Drawing;
```

10.3.2 - Définition

```
private:
    /// <summary>
    /// Variable nécessaire au concepteur.
    /// </summary>
    System::ComponentModel::Container ^components;
    Graphics ^ graph; //Définition de l'objet graph
```

10.3.3 - Principales méthodes

CreateGraphics () : Création de l'objet Graphics. A exécuter en général au chargement de la fenêtre.

```
ex : graph = CreateGraphics;
```

DrawImage(Image, Int32, Int32): Dessine l'image spécifiée, en utilisant sa taille physique d'origine, à l'emplacement indiqué par une paire de coordonnées.

```
ex :
// Create image.
Image ^ newImage = Image::FromFile( "SampImag.jpg" );

// Create coordinates for upper-left corner of image.
int x = 100;
int y = 100;

// Draw image to screen.
graphe->DrawImage( newImage, x, y );
```

DrawArc(Pen^ pen, int x, int y, int width, int height, int startAngle, int sweepAngle): Dessine un arc représentant une partie d'une ellipse spécifiée par une paire de coordonnées, une largeur et une hauteur.

```
ex :
// Create pen.
Pen^ blackPen = gnew Pen( Color::Black, 3.0f );
// Create coordinates of rectangle to bound ellipse.
int x = 40;
int y = 40;
int width = 100;
int height = 200;
// Create start and sweep angles on ellipse.
int startAngle = 45;
int sweepAngle = 270;

// Draw arc to screen.
graph->DrawArc( blackPen, x, y, width, height, startAngle, sweepAngle );
```

Clear(Color color): Efface l'intégralité de la surface de dessin et la remplit avec la couleur d'arrière-plan spécifiée.

Color est une structure qui représente la couleur d'arrière-plan de la surface de dessin. Une couleur est représentée par son code ARVB (alpha, rouge, vert, bleu).

```
ex :
//Clear screen with teal background.
graphe->Clear(Color::Teal)

//Clear screen with default form background.
graphe->Clear(this->BackColor);
```

10.4 - La classe Bitmap

Encapsule une bitmap GDI+, composée des données de pixels d'une image graphique et de ses attributs.

10.4.1 - Espace de nom

```
Using namespace System::Drawing;
```

10.4.2 - Définition

```
private:
/// <summary>
/// Variable nécessaire au concepteur.
/// </summary>
System::ComponentModel::Container ^components;
Bitmap ^ bmp; //Définition de l'objet bmp
```

10.4.3 - Constructeur

Bitmap(String^ filename) : Initialise une nouvelle instance de la classe Bitmap à partir du fichier spécifié.

```
ex :
String ^ strPicture= strPicture = openFileDialog->FileName;
bmp = gcnew Bitmap(strPicture);
```

10.4.4 - Principales propriétés

Height : Obtient la hauteur du Bitmap, en pixels.

Width : Obtient la hauteur du Bitmap, en pixels.

10.4.5 - Principales méthodes

GetPixel(int x, int y) : Obtient la couleur du pixel spécifié dans le Bitmap. Retourne une structure Color représentant la couleur du pixel spécifié. Une couleur est représentée par son code ARVB (alpha, rouge, vert, bleu).

```
ex :
for ( int col=0 ; col<bmp->Width ; col++)
{
    for ( int row=0 ; row<bmp->Height ; row++)
    {
        // Rechercher les pixel Rouges
        if( bmp->GetPixel(col,row).R==255 &&
            bmp->GetPixel(col,row).G==0 &&
            bmp->GetPixel(col,row).B==0)
        {
            //Do something
        }
    }
}
```

SetPixel(int x, int y) : Définit la couleur du pixel spécifié dans le Bitmap. Une couleur est représentée par son code ARVB (alpha, rouge, vert, bleu).

```
ex :
for ( int col=0 ; col<bmp->Width ; col++)
{
    for ( int row=0 ; row<bmp->Height ; row++)
    {
        // Rechercher les pixel Rouges et les changer en Vert
        if( bmp->GetPixel(col,row).R==255 &&
            bmp->GetPixel(col,row).G==0 &&
            bmp->GetPixel(col,row).B==0)
        {
            bmp->SetPixel(col,row).R==0;
            bmp->SetPixel(col,row).G==255;
            bmp->SetPixel(col,row).B==0;
        }
    }
}
```